

Azure Kubernetes Service (AKS) troubleshooting documentation

Welcome to Azure Kubernetes Service troubleshooting. These articles explain how to determine, diagnose, and fix issues that you might encounter when you use Azure Kubernetes Service (AKS). In the navigation pane on the left, browse through the article list or use the search box to find issues and solutions.

Learn how to use AKS with these quickstarts, tutorials, and samples

HOW-TO GUIDE

[AKS documentation](#)

Troubleshoot create operations

HOW-TO GUIDE

[Basic troubleshooting](#)

[K8SAPIServerConnFailVMExtensionError error \(51\)](#)

[K8SAPIServerDNSLookupFailVMExtensionError error \(52\)](#)

[MissingSubscriptionRegistration error](#)

[OutboundConnFailVMExtensionError error \(50\)](#)

[RequestDisallowedByPolicy error](#)

[SubnetIsFull error](#)

[SubscriptionRequestsThrottled error \(429\)](#)

Troubleshoot upgrade operations

HOW-TO GUIDE

[Upgrade fails because of NSG rules](#)

[PodDrainFailure error](#)

[PublicIPCountLimitReached error](#)

[QuotaExceeded error](#)

[SubnetIsFull error during an upgrade](#)

[Upgrade issues with Gen2 VMs on Windows AKS cluster](#)

Troubleshoot delete operations

HOW-TO GUIDE

[CannotDeleteLoadBalancerWithPrivateLinkService error](#)

[InUseRouteTableCannotBeDeleted error](#)

[LoadBalancerInUseByVirtualMachineScaleSet error](#)

[PublicIPAddressCannotBeDeleted error](#)

[TooManyRequestsReceived error](#)

Troubleshoot scale operations

HOW-TO GUIDE

[Common issues with running or scaling large AKS clusters](#)

[Failed to fix node group sizes](#)

Troubleshoot start operations

HOW-TO GUIDE

[Basic troubleshooting](#)

[K8SAPIServerConnFailVMExtensionError error \(51\)](#)

[K8SAPIServerDNSLookupFailVMExtensionError error \(52\)](#)

[OutboundConnFailVMExtensionError error \(50\)](#)

Troubleshoot node not ready

HOW-TO GUIDE

[Basic troubleshooting](#)

[Custom script extension errors](#)

[Expired certificates](#)

[Node not ready after being healthy](#)

[Node not ready but then recovers](#)

[Troubleshoot node auto-repair errors](#)

Cannot connect to application hosted on AKS cluster

HOW-TO GUIDE

[Basic troubleshooting](#)

[Custom NSG blocks traffic](#)

[Failures in the "az aks command invoke" command](#)

[Get and analyze HTTP response codes](#)

[Intermittent timeouts or server issues](#)

Cannot connect to AKS cluster

HOW-TO GUIDE

[Basic troubleshooting](#)

[Client IP address can't access API server](#)

[Config file isn't available when connecting](#)

[Tunnel connectivity issues](#)

[User can't get cluster resources](#)

Data collection guide

HOW-TO GUIDE

[Capture real-time system insights from cluster](#)

[Capture TCP dump from Linux node](#)

[Capture TCP dump from Windows node](#)

[Capture TCP packets from pod](#)

[Capture container dump from Windows node](#)

Troubleshoot AKS cluster performance issues



HOW-TO GUIDE

[Identify high CPU consuming containers](#)

[Identify memory saturation in AKS clusters](#)

Troubleshoot common issues



[Troubleshoot common issues with Azure Linux container hosts on AKS](#)

[Troubleshoot DNS resolution problems on AKS clusters](#)

Basic troubleshooting of AKS cluster creation issues

Article • 09/09/2024

This article outlines the basic troubleshooting methods to use if you can't create or deploy a Microsoft Azure Kubernetes Service (AKS) cluster successfully.

Prerequisites

- [Azure CLI](#) (version 2.0.59 or a later version).
- The Kubernetes [kubectl](#) tool. To install kubectl by using Azure CLI, run the [az aks install-cli](#) command.

View errors from Azure CLI

If the operation fails when you try to create clusters by using Azure CLI, the output displays error information. Here's a sample Azure CLI command and output:

```
Azure CLI

# Create a cluster specifying subnet

az aks create --resource-group myResourceGroup
--name MyManagedCluster \
--load-balancer-sku standard \
--vnet-subnet-id /subscriptions/<subscriptions-
id>/resourceGroups/myResourceGroup/providers/Microsoft.Network/virtualNetwor
ks/aks_demo_vnet/subnets/AKS
```

Sample of output:

```
Output

It is highly recommended to use USER assigned identity (option --assign-
identity)when you want to bring you own subnet, which will have no latency
for the role assignment to take effect. When you SYSTEM assigned identity,
azure-cli will grant Network Contributor role to the system assigned
identity after the cluster is created, and the role assignment will take
some time to take effect, see https://learn.microsoft.com/azure/aks/use-
managed-identity, proceed to create cluster with system assigned identity?
(y/N): y` 

(ControlPlaneAddOnsNotReady) Pods not in Running status: konnectivity-agent-
```

```
67f7f5554f-nsw2g,konnectivity-agent-8686cb54fd-xlsgk,metrics-server-6bc97b47f7-dfhbr,coredns-845757d86-7xjqb,coredns-autoscaler-5f85dc856b-mxkrj
```

You can identify the error code and error message from the output. In this case, they are:

- **Error code:** ControlPlaneAddOnsNotReady
- **Error message:** Pods not in Running status: konnectivity-agent-67f7f5554f-nsw2g,konnectivity-agent-8686cb54fd-xlsgk,metrics-server-6bc97b47f7-dfhbr,coredns-845757d86-7xjqb,coredns-autoscaler-5f85dc856b-mxkrj.

These descriptions often contain details of what went wrong in the cluster creation, and they link to articles that contain even more details. Additionally, you can use our troubleshooting articles as a reference, based on the errors that the Azure CLI operation produces.

View error details in the Azure portal

To investigate the AKS cluster creation errors in the [Azure portal](#), open the **Activity Log**. You can filter the results to fit your needs. To do this, select **Add Filter** to add more properties to the filter.



On the **Activity Log** page, locate log entries in which the **Operation name** column shows **Create or Update Managed Cluster**.

The **Event initiated by** column shows the user who performed the operation, which could be a work account, a school account, or an [Azure managed identity](#).

If the operation is successful, the **Status** column value is **Accepted**. You'll also see suboperation entries for the creation of the cluster components, such as the following operation names:

- Create or Update Route Table
- Create or Update Network Security Group
- Update User Assigned Identity Create
- Create or Update Load Balancer
- Create or Update Public Ip Address
- Create role assignment
- Update resource group

In these suboperation entries, the **Status** value is **Succeeded** and the **Event initiated by** field is set to **AzureContainerService**.

The screenshot shows the Azure Log Analytics Activity log interface. On the left, there's a sidebar with navigation links like 'Activity log' (highlighted with a red arrow), 'Access control (IAM)', 'Tags', etc. The main area displays a table of activity logs with columns: Operation name, Status, Time, Time stamp, Subscription, and Event initiated by. There are three items listed:

Operation name	Status	Time	Time stamp	Subscription	Event initiated by
Create or Update Managed Cluster	Succeeded	a day ago	Thu Aug 29 ...		id
Create or Update Managed Cluster	Started	a day ago	Thu Aug 29 ...		id
Create or Update Managed Cluster	Accepted	a day ago	Thu Aug 29 ...		id

What if an error occurred instead? In that case, the **Status** value is **Failed**. Unlike in the operations to create cluster components, you must expand the failed suboperation entries to review them. Typical suboperation names are policy actions, such as '**audit**' **Policy action** and '**auditIfNotExists**' **Policy action**. Not all suboperations necessarily fail together. You can expect to see some of them succeed.

Select one of the failed suboperations to further investigate it. Select the **Summary**, **JSON**, and **Change History** tabs to troubleshoot the issue. The **JSON** tab contains the output text for the error in JSON format, and it usually provides the most helpful information.

The screenshot shows the Azure Log Analytics Activity log interface for a failed operation. The left sidebar has 'Activity log' selected. The main table shows one item with a status of 'Failed'. Expanding this item reveals three suboperations: 'Create or Update Agent Pool' (Failed), 'Create or Update Agent Pool' (Started), and 'Create or Update Agent Pool' (Accepted). The 'JSON' tab on the right displays the detailed log for the failed suboperation:

```
58     "value": "microsoft.containerservice/managedclusters/agentpools/write",
59     "localizedValue": "Create or Update Agent Pool"
60   },
61   "resourceGroupName": "rgaks",
62   "resourceProviderName": {
63     "value": "microsoft.containerservice",
64     "localizedValue": "microsoft.containerservice"
65   },
66   "resourceType": {
67     "value": "microsoft.containerservice/providers/microsoft.containers/agentpools",
68     "localizedValue": "microsoft.containerservice/providers/microsoft.containers/agentpools"
69   },
70   "resourceId": "/subscriptions//resourceGroups/rgaks/providers/microsoft.containerservice/managedClusters/clusterName/agentPools/agentPoolName"
71   ...
72   "status": {
73     "value": "Failed",
74     "localizedValue": "Failed"
75   }
76 }
```

Here's an example of the detailed log in JSON format:

The screenshot shows a JSON code editor with the word 'JSON' at the top. The code is as follows:

```
{
  "status": {
    "value": "Failed",
    "localizedValue": "Failed"
  },
  "subStatus": {
    "value": "",
    "localizedValue": ""
  },
  "submissionTimestamp": "2024-08-30T10:06:07Z",
  "subscriptionId": "<subscriptionId>",
  "tenantId": "<tenantId>",
  "properties": {
```

```
        "statusMessage": "{\"status\":\"Failed\",\"error\":\n        {\"code\":\"ResourceOperationFailure\",\"message\":\"The resource operation\n        completed with terminal provisioning state 'Failed'.\",\"details\":\n        [{\"code\":\"VMExtensionProvisioningError\",\"message\":\"Unable to\n        establish outbound connection from agents, please see\n        https://learn.microsoft.com/en-us/troubleshoot/azure/azure-kubernetes/error-\n        code-outboundconnfailvmextensionerror and https://aka.ms/aks-required-ports-\n        and-addresses for more information.\"]}]}}},\n    }\n}
```

View cluster insights

Was the cluster created in the Azure portal, and is it visible there? If this is true, you can generate cluster insights that will help you troubleshoot. To access this feature, follow these steps:

1. In [the Azure portal](#), search for and select **Kubernetes services**.
2. Select the name of your AKS cluster.
3. In the navigation pane of the AKS cluster page, select **Diagnose and solve problems**.
4. In the **Diagnose and solve problems** page, select the **Cluster insights** link. The cluster insights tool analyzes your cluster, and then provides a list of its findings in the **Observations and Solutions** section of the **Cluster Insights** page.
5. Select one of the findings to view more information about a problem and its possible solutions.

View resources in the Azure portal

In the Azure portal, you might want to view the resources that were created when the cluster was built. Typically, these resources are in a resource group whose name begins in *MC_*. The managed cluster resource group might have a name such as **MC_MyResourceGroup_MyManagedCluster_<location-code>**. However, the name may be different if you built the cluster by using a custom-managed cluster resource group.

To find the resource group, search for and select **Resource groups** in the Azure portal, and then select the resource group in which the cluster was created. The resource list is shown in the **Overview** page of the resource group.

Warning

We recommend that you don't modify resources in the *MC_* resource group. This action might adversely affect your AKS cluster.

To review the status of a virtual machine scale set, you can select the scale set name within the list of resources for the resource group. It might have a **Name** value that resembles `aks-nodepool1-12345678-vmss`, and it a **Type** value of **Virtual machine scale set**. The status of the scale set appears at the top of the node pool's **Overview** page, and more details are shown in the **Essentials** heading. If deployment was unsuccessful, the displayed status is **Failed**.

For all resources, you can review details to gain a better understanding about why the deployment failed. For a scale set, you can select the **Failed** status text to view details about the failure. The details are in a row that contains **Status**, **Level**, and **Code** columns. The following example shows a row of column values.

[] Expand table

Column	Example value
Status	Provisioning failed
Level	Error
Code	ProvisioningState/failed/VMExtensionProvisioningError

Select the row to see the **Message** field. This contains even more information about that failure. For example, the **Message** field for the example row begins in the following text:

VM has reported a failure when processing extension 'vmssCSE'. Error message:
"Enable failed: failed to execute command: command terminated with exit status=50
[stdout] [stderr] 0 0 0 --:

Armed with this information, you can conclude that the VMs in the scale set failed and generated exit status 50.

(!) Note

If the cluster deployment didn't reach the point at which these resources would have been created, you might not be able to review the managed cluster resource group in the Azure portal.

Use Kubectl commands

For another option to help troubleshoot errors on your cluster, use kubectl commands to get details about the resources that were deployed in the cluster. To do this, first sign in to your AKS cluster:

Azure CLI

```
az aks get-credentials --resource-group MyResourceGroup --name  
MyManagedCluster
```

Depending on the type of failure and when it occurred, you might not be able to sign in to your cluster to get more details. But if your cluster was created and appears in the Azure portal, you should be able to sign in and run kubectl commands.

View cluster nodes (kubectl get nodes)

To determine the state of the cluster nodes, view the nodes by running the [kubectl get nodes](#) command. In this example, no nodes are reporting in the cluster:

Console

```
$ kubectl get nodes  
  
No resources found
```

View pods in the system namespace (kubectl get pods)

Viewing the pods in the kube-system namespace is also a good way to troubleshoot your issue. This method lets you view the status of the Kubernetes system pods. In this example, we enter the `kubectl get pods` command:

Console

```
$ kubectl get pods -n kube-system  
NAME                               READY   STATUS    RESTARTS   AGE  
coredns-845757d86-7xjqb           0/1     Pending   0          78m  
coredns-autoscaler-5f85dc856b-mxkrj 0/1     Pending   0          77m  
konnektivity-agent-67f7f554f-nsw2g  0/1     Pending   0          77m  
konnektivity-agent-8686cb54fd-xlsgk 0/1     Pending   0          65m  
metrics-server-6bc97b47f7-dfhbr   0/1     Pending   0          77m
```

Describe the status of a pod (kubectl describe pod)

By describing the status of the pods, you can view the configuration details and any events that have occurred on the pods. Run the [kubectl describe pods](#) command:

```
Console

$ kubectl describe pod coredns-845757d86-7xjqb -n kube-system
Name:           coredns-845757d86-7xjqb
Namespace:      kube-system
Priority:      2000001000
Priority Class Name: system-node-critical
Node:          <none>
Labels:         k8s-app=kube-dns
                kubernetes.io/cluster-service=true
                pod-template-hash=845757d86
                version=v20
...
Events:
  Type     Reason        Age           From            Message
  ----     -----        ---          ----           -----
  Warning  FailedScheduling  24m (x1 over 25m)  default-scheduler  no nodes
available to schedule pods
  Warning  FailedScheduling  29m (x57 over 84m)  default-scheduler  no nodes
available to schedule pods
```

In the command output, you can see that the pod can't deploy to a node because no nodes are available.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Troubleshoot Container Network Interface download failures

07/08/2025

This article discusses how to identify and resolve the `CniDownloadTimeoutVMExtensionError` error code (also known as error code `ERR_CNI_DOWNLOAD_TIMEOUT`, error number 41) or the `WINDOWS_CSE_ERROR_DOWNLOAD_CNI_PACKAGE` error code (error number 35) that occurs when you try to create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- The [Curl](#) command-line tool
- Network access from the same environment where AKS nodes will be deployed (same VNet, firewall rules, etc.)

Symptoms

When you try to create a Linux-based AKS cluster, you receive the following error message:

Output

```
Message: We are unable to serve this request due to an internal error
SubCode: CniDownloadTimeoutVMExtensionError;
Message="VM has reported a failure when processing extension 'vmssCSE'.
Error message: "Enable failed: failed to execute command: command terminated with
exit status=41\n[stdout]\n{
"ExitCode": "41",
```

When you try to create a Windows-based AKS cluster, you receive the following error message:

Output

```
Message="VM has reported a failure when processing extension 'vmssCSE' (publisher
'Microsoft.Compute' and type 'CustomScriptExtension').
Error message: 'Command execution finished, but failed because it returned a non-
zero exit code of: '1''. The command had an error output of: 'ExitCode: |35|,
Output: |WINDOWS_CSE_ERROR_DOWNLOAD_CNI_PACKAGE|, Error: |Failed in downloading
\r\nhttps://acs-mirror.azureedge.net/azure-cni/v1.4.56/binaries/azure-vnet-cni-
overlay-windows-amd64-v1.4.56.zip.
Error: \r\nUnable to connect to the r|\r\nAt line:1 ...
For more information, check the instance view by executing Get-AzVmssVm or Get-
AzVm (https://aka.ms/GetAzVm). These commands can be executed using CloudShell
```

```
(https://aka.ms/CloudShell)'. More information on troubleshooting is available at https://aka.ms/VMExtensionCSEWindowsTroubleshoot.
```

Cause

Your cluster nodes can't connect to the endpoint that's used to download the Container Network Interface (CNI) libraries. In most cases, this issue occurs because a network virtual appliance is blocking Secure Sockets Layer (SSL) communication or an SSL certificate.

Solution

Run a Curl command to verify that your nodes can download the binaries:

First, attempt a test download of the Azure CNI package for Linux from the official mirror endpoint.

Bash

```
curl -I https://acs-mirror.azureedge.net/cni/azure-vnet-cni-linux-amd64-v1.0.25.tgz
```

Results:

Output

```
HTTP/2 200
content-length: 970752
content-type: application/x-gzip
last-modified: Wed, 22 Jun 2022 00:00:00 GMT
etag: "0x8DA53F1234567"
server: ECACC (dab/4B9E)
x-cache: HIT
cache-control: public, max-age=86400
accept-ranges: bytes
date: Thu, 05 Jun 2025 00:00:00 GMT
```

This command checks if the endpoint is reachable and returns the HTTP headers. If you see a `200 OK` response, it indicates that the endpoint is accessible.

Next, attempt a download with validation and save the file locally for further troubleshooting. This will help determine if SSL or outbound connectivity is correctly configured.

Bash

```
# Create a temporary directory for testing
mkdir -p /tmp/cni-test

# Download the CNI package to the temp directory
curl -L --fail https://acs-mirror.azureedge.net/cni/azure-vnet-cni-linux-amd64-
v1.0.25.tgz --output /tmp/cni-test/azure-vnet-cni-linux-amd64-v1.0.25.tgz && echo
"Download successful" || echo "Download failed"
```

Results:

Output

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current
Dload	Upload	Total	Total	Spent	Left	Speed	Speed
100	6495k	100	6495k	0	0	8234k	8230k
Download successful							

Verify the downloaded file:

Bash

```
ls -la /tmp/cni-test/
file /tmp/cni-test/azure-vnet-cni-linux-amd64-v1.0.25.tgz
```

Results:

Output

```
total 6500
drwxr-xr-x 2 user user    4096 Jun 20 10:30 .
drwxrwxrwt 8 root root    4096 Jun 20 10:30 ..
-rw-r--r-- 1 user user 6651392 Jun 20 10:30 azure-vnet-cni-linux-amd64-v1.0.25.tgz

/tmp/cni-test/azure-vnet-cni-linux-amd64-v1.0.25.tgz: gzip compressed data, from
Unix, original size modulo 2^32 20070400
```

Clean up the test files:

Bash

```
rm -rf /tmp/cni-test/
```

If you can't download these files, make sure that traffic is allowed to the downloading endpoint. For more information, see [Azure Global required FQDN/application rules](#).

References

- General troubleshooting of AKS cluster creation issues

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the ERR_VHD_FILE_NOT_FOUND error code (65)

Article • 02/13/2025

This article discusses how to identify and resolve the `ERR_VHD_FILE_NOT_FOUND` error code (error code number 65) that occurs when you try to create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to create an AKS cluster, you receive the following error message:

`VMExtensionProvisioningError: VM has reported a failure when processing extension 'vmssCSE'.`

Error message: "Enable failed: failed to execute command: command terminated with exit status=65

Cause

Under rare circumstances, the 65 exit code for the Azure Virtual Machine Scale Set custom script extension (`vmssCSE`) might happen instead of the following error codes:

[+] Expand table

Error code name	Error code number
<code>OutboundConnFailVMExtensionError</code>	50
<code>K8SAPIServerConnFailVMExtensionError</code>	51
<code>K8SAPIServerDNSLookupFailVMExtensionError</code>	52

This error occurs if a connectivity issue exists between your AKS cluster and the required Azure endpoints, such as `mcr.microsoft.com` or `acs-mirror.azureedge.net`.

Solution

Review [outbound network and FQDN rules for Azure Kubernetes Service \(AKS\) clusters](#) and make sure that all API services FQDN are allowed.

For detailed troubleshooting steps, refer to the troubleshooting guides in the following articles:

- Troubleshoot the OutboundConnFailVMExtensionError error code (50)
- Troubleshoot the K8SAPIServerConnFailVMExtensionError error code (51)
- Troubleshoot the K8SAPIServerDNSLookupFailVMExtensionError error code (52)

References

- [General troubleshooting of AKS cluster creation issues](#)
- [Outbound network and FQDN rules for Azure Kubernetes Service \(AKS\) clusters](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

Troubleshoot the InvalidParameter error

Article • 11/27/2024

This article discusses how to identify and resolve the `InvalidParameter` error that occurs when you try to create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- [Azure CLI](#), version 2.0.81 or a later version. If Azure CLI is already installed, you can find the version number by running `az --version`.

Symptoms

When you create an AKS cluster, the provided configurations are usually validated before the cluster is created. However, on rare occasions, a parameter passes validation before the AKS cluster is created but causes errors when the resources for the cluster are created. Errors that are related to invalid parameters might resemble the following examples:

- Scenario: The selected VM size is not available

```
Code="InvalidParameter"

Message="**The requested VM size Standard_D4s_v3 is not available in
the current region. The sizes available in the current region are:
ExtraSmall_Internal, Small_Internal, Medium_Internal, Large_Internal,
ExtraLarge_Internal, Standard_DC2as_v5, Standard_DC4as_v5,
Standard_DC8as_v5, Standard_DC16as_v5, Standard_DC32as_v5,
Standard_DC48as_v5, Standard_DC64as_v5, Standard_DC96as_v5,
Standard_DC2ads_v5, Standard_DC4ads_v5, Standard_DC8ads_v5,
Standard_DC16ads_v5, Standard_DC32ads_v5, Standard_DC48ads_v5,
Standard_DC64ads_v5, Standard_DC96ads_v5, Standard_EC2as_v5,
Standard_EC4as_v5, Standard_EC8as_v5, Standard_EC16as_v5,
Standard_EC20as_v5, Standard_EC32as_v5, Standard_EC48as_v5,
Standard_EC64as_v5, Standard_EC96as_v5, Standard_EC96ias_v5,
Standard_EC2ads_v5, Standard_EC4ads_v5, Standard_EC8ads_v5,
Standard_EC16ads_v5, Standard_EC20ads_v5, Standard_EC32ads_v5,
Standard_EC48ads_v5, Standard_EC64ads_v5, Standard_EC96ads_v5,
Standard_EC96iads_v5.\r\nFind out more on the available VM sizes in
each region at <https://aka.ms/azureregions>."
```

```
Target="vmSize"
```

- Scenario: Cluster names are unavailable or conflict with Azure reserved values
 - Example 1

```
Code="InvalidParameter"

Message="The value of parameter name is invalid. Error details:
"omsagent-aks-dev-microsoft" managed cluster name is invalid because
'MICROSOFT' and 'WINDOWS' can't be used as either a whole word or a
substring in the name.. Please see https://aka.ms/aks-naming-rules
for more details."
```

- Example 2

```
Message="The value of parameter name is invalid. Error details:
"login" managed cluster name is invalid because 'LOGIN' and 'XBOX'
can't be used at the start of a resource name, but can be used later
in the name.. Please see https://aka.ms/aks-naming-rules for more
details."
```

- Example 3

```
Message=" The value of parameter name is invalid. Error details:
"azure" managed cluster name is invalid because it is reserved..
Please see https://aka.ms/aks-naming-rules for more details.
Target: name"
```

Cause

This issue occurs because one of the following conditions is true:

- The Azure Virtual Machine SKU isn't available in the selected region.
- The service principal is invalid.
- A virtual network, subnet, or route table is invalid.
- An Azure CLI parameter is invalid.
- The value of parameter name is unavailable or reserved by Azure.

There might also be other reasons that your cluster creation attempt failed.

Solution

In the following table, follow the link for the appropriate troubleshooting step.

[Expand table](#)

Troubleshooting step	Reference link
Check whether the SKU is available	Resolve errors for SKU not available
Verify that the service principal is valid	Service principals together with AKS
Verify that any commands that were used to create the cluster are valid	az aks (Azure CLI reference)
Verify that any custom network resources that were used to create the cluster are valid	Configure Azure CNI networking in AKS and Customize cluster egress with a user-defined route
Avoid using unavailable or Azure-reserved values for names	Refer to the error messages provided

More information

- General troubleshooting of AKS cluster creation issues

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

Troubleshoot the InvalidResourceReference error code

Article • 04/10/2024

This article discusses how to identify and resolve the `InvalidResourceReference` errors that may occur when you try to create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster or update an AKS cluster.

Symptom 1

When you try to create an AKS cluster, you receive the following error message:

```
Code="InvalidResourceReference"

Message="Resource
/subscriptions/<subscription-id-
guid>/resourceGroups/MyResourceGroup/providers/Microsoft.Network/virtualN
etworks/vnet-otcom/subnets/Subnet-AKS
referenced by resource
/subscriptions/<subscription-id-
guid>/resourceGroups/MC_MyResourceGroup_MyCluster-
AKS_JAPANEAST/providers/Microsoft.Compute/virtualMachineScaleSets/aks-
nodepool-vmss
was not found. Please make sure that the referenced resource exists, and that both
resources are in the same region."
Details=[]
```

Cause 1

Here are the possible causes of this issue:

- A mismatch exists between resources in different regions.

The example in [Symptom 1](#) shows that the virtual network and the virtual machine scale set aren't in the same region. Because the resources are in different regions, it's impossible to create the scale set instance.

- The referenced resource has been manually modified or deleted.

Solution 1

If a mismatch exists between resources in different regions, review the resources to make sure that they're in the same region. In this example, either modify the region where the AKS cluster is being built, or create a new virtual network in the same region.

If the referenced resource has been manually modified or deleted, it might be difficult to resolve this issue because it's unsupported to manually modify the underlying IaaS resources in the *MC_* resource group. A possible solution might be to recreate the deleted resource, reassociate it with the VMSS, and then trigger an update on the AKS cluster. However, as this is an unsupported scenario, the success of this solution can't be guaranteed.

Symptom 2

When you try to update an AKS cluster, you receive the following error message:

```
Code="InvalidResourceReference"
Message="Resource
/subscriptions/<subscription-id-
guid>/resourceGroups/MC_MyResourceGroup/providers/Microsoft.Network/loadBal-
ancers/kubernetes/frontendIPConfigurations/<frontendIP_ID> referenced by
resource /subscriptions/<subscription-id-
guid>/resourceGroups/MC_MyResourceGroup/providers/Microsoft.Network/loadBal-
ancers/kubernetes/loadBalancingRules/<frontend_IP_rule> was not found. Please
make sure that the referenced resource exists, and that both resources are in the
same region."
Message="Resource
Details=[]
```

Cause 2

This issue might occur if the default outbound rule "aksOutboundRule" on the load balancer is manually modified. This unexpected modification typically occurs when the outbound IP is updated if you update the cluster without the `load-balancer-outbound-ips` parameter.

Solution 2

Rerun the `az aks update` command with the `load-balancer-outbound-ips` parameter to update your cluster. Use the resource ID of the public IP as the parameter value. For more information, see [Update the cluster with your own outbound public IP](#).

More information

[General troubleshooting of AKS cluster creation issues](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Known issues: Custom kubelet configuration on Windows nodes in AKS

Article • 04/10/2024

This article discusses known issues that affect custom kubelet configurations on Microsoft Windows nodes in the Azure Kubernetes Service (AKS). For more information about this feature, see [Customize node configuration for AKS node pools](#).

Log sizes exceed the specified maximum during aggressive log writing

On a Windows virtual machine (VM), log sizes grow beyond the value of the `container-log-max-size` setting if a container is writing to the log aggressively. During heavy log writing periods, the log file grows too quickly for [log rotation](#) to occur before the `container-log-max-size` setting limit is exceeded.

If several pods are aggressively writing to the log, the log size can grow to dozens of gibibytes (GiBs) before the log is rotated (compressed and replaced), even if the maximum size is only in the dozens of mebibytes (MiBs).

For more information, see Kubernetes [GitHub issue 110630](#), "Kubelet does not respect `container-log-max-size` on time, during heavy log writes from container."

Affected versions

The excessive log size issue applies to all versions of Kubernetes.

Kubelet log file compression fails

On a Windows VM, when the kubelet tries to compress a log file into a `.gz` archive format, it stops responding during the final step of the process (trying to rename the archive before it closes the file).

For more information, see Kubernetes [GitHub issue 111548](#), "Kubelet log compression fails on Windows."

Affected versions

The kubelet log file compression issue applies to all versions of Kubernetes that are older than version 1.23. It applies also to certain early versions of Kubernetes 1.23 and 1.24, as shown in the following table. The log file compression issue is fixed for Kubernetes version 1.25.0 (in Kubernetes [GitHub pull request 111549](#)) and all subsequent versions of Kubernetes.

[+] [Expand table](#)

Kubernetes x.y version	Versions that the known issue applies to	GitHub fix pull request number on Kubernetes
1.24	All versions before 1.24.7	112482
1.23	All versions before 1.23.13	112483

For more information, see the [January 29, 2023 release](#) of the AKS changelog.

Custom OS configurations fail

Symptoms

A custom OS configuration doesn't get applied.

Cause

This issue occurs if you try to apply the custom OS configuration on a Windows node pool. Currently, OS configurations aren't supported on Windows node pools. These configurations work only on Linux node pools.

Workaround

Apply the custom OS configuration at the cluster level or node pool level for a Linux node pool. To check whether the custom node configuration is in use, see [Confirm settings have been applied](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Troubleshoot the K8SAPIServerConnFailVMExtensionError error code (51)

Article • 03/12/2025

This article discusses how to identify and resolve the `K8SAPIServerConnFailVMExtensionError` error (also known as error code `ERR_K8S_API_SERVER_CONN_FAIL`, error number 51) that occurs when you try to start or create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- The [Netcat](#) (nc) command-line tool

Symptoms

When you try to start or create an AKS cluster, you receive the following error message:

Unable to establish connection from agents to Kubernetes API server, please see <https://aka.ms/aks-required-ports-and-addresses> for more information.

Details: Code="VMExtensionProvisioningError"

Message="VM has reported a failure when processing extension 'vmssCSE'.

Error message: "Enable failed: failed to execute command: command terminated with exit status=51\n[stdout]\n{

"ExitCode": "51",

"Output": "Thu Oct 14 18:07:37 UTC 2021,aks-nodepool1-18315663-vmss000000\nConnection to

Cause

Your cluster nodes can't connect to your cluster API server pod.

Solution

Run a Netcat command to verify that your nodes can resolve the cluster's fully qualified domain name (FQDN):

```
shell
```

```
nc -vz <cluster-fqdn> 443
```

If you're using egress filtering through a firewall, make sure that traffic is allowed to your cluster FQDN.

In rare cases, the firewall's outbound IP address can be blocked if you've authorized IP addresses that are enabled on your cluster. In this scenario, you must add the outbound IP address of your firewall to the list of authorized IP ranges for the cluster. For more information, see [Secure access to the API server using authorized IP address ranges in AKS](#).

More information

- [General troubleshooting of AKS cluster creation issues](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

Troubleshoot the K8SAPIServerDNSLookupFailVMExtensionError error code (52)

07/08/2025

This article discusses how to identify and resolve the `K8SAPIServerDNSLookupFailVMExtensionError` error (also known as error code `ERR_K8S_API_SERVER_DNS_LOOKUP_FAIL`, error number 52) that occurs when you try to start or create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- The [nslookup](#) DNS lookup tool for Windows nodes or the [dig](#) tool for Linux nodes.
- [Azure CLI](#), version 2.0.59 or a later version. If Azure CLI is already installed, you can find the version number by running `az --version`.

Symptoms

When you try to start or create an AKS cluster, you receive the following error message:

Agents are unable to resolve Kubernetes API server name. It's likely custom DNS server is not correctly configured, please see <https://aka.ms/aks/private-cluster#hub-and-spoke-with-custom-dns> for more information.

Details: Code="VMExtensionProvisioningError"

Message="VM has reported a failure when processing extension 'vmssCSE'.

Error message: "Enable failed: failed to execute command: command terminated with exit status=52\n[stdout]\n{

"ExitCode": "52",

"Output": "Fri Oct 15 10:06:00 UTC 2021,aks- nodepool1-36696444-vmss000000\\nConnection to mcr.microsoft.com 443 port [tcp/https]

Cause

The cluster nodes can't resolve the cluster's fully qualified domain name (FQDN) in Azure DNS. Run the following DNS lookup command on the failed cluster node to find DNS resolutions that are valid.

[+] [Expand table](#)

Node OS	Command
Linux	<code>dig <cluster-fqdn></code>
Windows	<code>nslookup <cluster-fqdn></code>

Solution

On your DNS servers and firewall, make sure that nothing blocks the resolution to your cluster's FQDN. Your custom DNS server might be incorrectly configured if something is blocking even after you run the `nslookup` or `dig` command and apply any necessary fixes. For help to configure your custom DNS server, review the following articles:

- [Create a private AKS cluster](#)
- [Private Azure Kubernetes service with custom DNS server ↗](#)
- [What is IP address 168.63.129.16?](#)

When you use a private cluster that has a custom DNS, a DNS zone is created. The DNS zone must be linked to the virtual network. This occurs after the cluster is created. Creating a private cluster that has a custom DNS fails during creation. However, you can restore the creation process to a "success" state by reconciling the cluster. To do this, run the `az resource update` command in Azure CLI, as follows:

Below, set your AKS cluster and resource group names, then run the update command to reconcile the cluster. The environment variables will make your resource names unique and are declared just before use.

Azure CLI

```
az resource update --resource-group $RESOURCE_GROUP_NAME \
--name $CLUSTER_NAME \
--namespace Microsoft.ContainerService \
--resource-type ManagedClusters
```

Results:

Output

```
{  
  "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-  
xxxxxxxxxxxx/resourceGroups/myResourceGroupxxx/providers/Microsoft.ContainerService/ManagedClusters/myAksClusterxxx",  
  "location": "eastus",  
  "name": "myAksClusterxxx",  
  "properties": {  
    // ...other properties...  
  },  
  "resourceGroup": "myResourceGroupxxx",  
  "type": "Microsoft.ContainerService/ManagedClusters"  
}
```

Also verify that your DNS server is configured correctly for your private cluster, as described earlier.

 **Note**

Conditional Forwarding doesn't support subdomains.

More information

- [General troubleshooting of AKS cluster creation issues](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the LinkedAuthorizationFailed error code

Article • 08/28/2024

This article discusses how to identify and resolve the `LinkedAuthorizationFailed` error that occurs when you try to create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to create an AKS cluster, you receive the following error message:

Reconcile VNet failed.

Details: VNetReconciler retry failed:

Category: ClientError; SubCode: LinkedAuthorizationFailed;

Dependency: Microsoft.Network/virtualNetworks; OrginalError:

Code="LinkedAuthorizationFailed"

Message="The client '12345678-1234-1234-1234-123456789098' with object id '123456789-1234-1234-1234-1234567890987' has permission to perform action 'Microsoft.Network/virtualNetworks/write' on scope '/subscriptions/<*subscription-id*-*guid*>/resourceGroups/MC_MyRG_westeurope/providers/Microsoft.Network/virtualNetworks/aks-vnet'; however, it does not have permission to perform action 'Microsoft.Network/ddosProtectionPlans/join/action' on the linked scope(s) '/subscriptions/<*subscription-id-guid*>/resourcegroups/ddos-protection-plan-rg/providers/microsoft.network/ddosprotectionplans/upmddosprotectionplan' or the linked scope(s) are invalid.";

AKSTeam: Networking, Retriable: false.

Cause

A service principal doesn't have permission to use a resource that's required for cluster creation.

Solution

Grant the service principal permissions to use the resource that's mentioned in the error message. The example output in the "Symptoms" section provides the following information.

[+] Expand table

Item	Value
Service principal	12345678-1234-1234-1234-123456789098
Resource	/subscriptions/<subscription-id-guid>/resourcegroups/ddos-protection-plan-rg/providers/microsoft.network/ddosprotectionplans/upmddosprotectionplan
Operation	Microsoft.Network/ddosProtectionPlans/join/action

For more information about how to grant permissions to the service principal, see [Assign Azure roles using the Azure portal](#).

More information

- General troubleshooting of AKS cluster creation issues

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Troubleshoot the MissingSubscriptionRegistration error code

Article • 09/16/2024

This article discusses how to identify and resolve the `MissingSubscriptionRegistration` error that occurs when you try to create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to deploy an AKS cluster, you receive the following error message:

```
Code="MissingSubscriptionRegistration"

Message="The subscription is not registered to use namespace
'Microsoft.OperationsManagement'. See https://aka.ms/rps-not-found for how
to register subscriptions."

Details=[

  "code": "MissingSubscriptionRegistration",

  "message": "The subscription is not registered to use namespace
'Microsoft.OperationsManagement'. See https://aka.ms/rps-not-found for how to
register subscriptions.",

  "target": "Microsoft.OperationsManagement"

]
```

Cause

You receive this error message for one of the following reasons:

- The required resource provider isn't registered for your subscription.
- The API version isn't supported for the resource type.
- The location isn't supported for the resource type.

Solution

To resolve this issue, follow the instructions in the "Solution" section of [Resolve errors for resource provider registration](#). Replace the existing namespace with the namespace that's shown in the error message. In the example in the "Symptoms" section, the namespace is `Microsoft.OperationsManagement`.

More information

- General troubleshooting of AKS cluster creation issues

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?



Troubleshoot the OperationNotAllowed or PublicIPCountLimitReached quota error

Article • 10/28/2024

This article describes how to identify and resolve the `OperationNotAllowed` or `PublicIPCountLimitReached` quota error that occurs when you try to create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to create an AKS cluster, you receive the following error message:

Code="OperationNotAllowed"

Message="Operation could not be completed as it results in exceeding approved standardEv3Family Cores quota.

Additional details - Deployment Model: Resource Manager,

Location: uksouth,

Current Limit: 200,

Current Usage: 200,

Additional Required: 8,

(Minimum) New Limit Required: 208.

Submit a request for Quota increase at

https://aka.ms/ProdportalCRP/#blade/Microsoft_Azure_Capacity/UsageAndQuota.ReactView/Parameters/<parameter-id> by specifying parameters listed in the 'Details' section for deployment to succeed. Please read more about quota limits at <https://learn.microsoft.com/azure/azure-supportability/per-vm-quota-requests>"

Or, you receive the following error message:

Reconcile standard load balancer failed.

Details: outboundReconciler retry failed: Category: ClientError; SubCode: PublicIPCountLimitReached;
Dependency: Microsoft.Network/PublicIPAddresses;
OrginalError: Code="PublicIPCountLimitReached"
Message="Cannot create more than 1000 public IP addresses for this subscription in this region." Details=[]; AKSTeam: Networking, Retriable: false.

Cause

You've exhausted the quota for a resource in an SKU's region.

Solution 1: Request a quota increase for your SKU

Typically, these error messages provide a link to create a support ticket to increase the quota for that SKU. In most cases, these requests are approved automatically, and the increased quota will be available in a short time.

If you don't have the link to request a quota increase, make the request in the Azure portal, as follows:

1. In the [Azure portal](#), search for and select **Quotas**.
2. Select **Microsoft.Compute** to view the list of quota records.
3. Use the **Location** list to filter for records from only your region.
4. In the specific record that's out-of-quota, select the pencil icon at the end of the record row.
5. In the **Request quota increase** pane, complete the **New limit** field, and then select **Submit**.

After your quota increase request is approved, you can retry the cluster creation operation.

Solution 2: Reprovision the cluster in another region or SKU

Alternatively, you can reprovision the cluster in a different region or SKU that has enough capacity for your cluster.

More information

- General troubleshooting of AKS cluster creation issues

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?



[Provide product feedback](#)

AKSOperationPreempted or AKSOperationPreemptedByDelete error when performing a new operation

Article • 10/28/2024

This article discusses how to identify and resolve the `AKSOperationPreempted` or `AKSOperationPreemptedByDelete` error that might occur when you try to perform a new operation but it has been preempted by another operation on a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to perform a new operation on an AKS cluster, you receive one of the following error messages:

- Code: "AKSOperationPreempted"
Message: "This operation has been preempted by another operation with ID <operation ID>"

This message indicates that the operation has been preempted by another operation, and the GUID for the new operation is given in the error message.

- Code: "AKSOperationPreemptedByDelete"
Message: "This operation has been preempted by Deleting operation."

This message indicates that the operation has been preempted by a delete operation.

Cause

This error usually occurs when an in-progress operation is interrupted by a subsequent operation that was issued before the in-progress operation is finished. The error will indicate the subsequent operation, which can be a delete or any other operation.

Solution

To resolve this issue, use one of the following methods. Once there are no operations running, you can try to run your operation again.

- Wait for the previous operation to complete.

You can check the status of the operation using the ID given in the error message with the following command:

Azure CLI

```
az aks operation show \
--resource-group myResourceGroup \
--name myCluster \
--operation-id "<operation-id>"
```

- Run an `abort` command to stop the previous operation.

For more information about how to abort an operation, see [Terminate a long running operation on an Azure Kubernetes Service \(AKS\) cluster](#).

More information

[General troubleshooting of AKS cluster creation issues](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

Troubleshoot the OutboundConnFailVMExtensionError error code (50)

07/24/2025

This article describes how to identify and resolve the `OutboundConnFailVMExtensionError` error (also known as error code `ERR_OUTBOUND_CONN_FAIL`, error number 50) that might occur if you create, start, scale, or upgrade a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- The [Netcat](#) (nc) command-line tool
- The [dig](#) command-line tool
- The Client URL ([cURL](#)) tool

Symptoms

When you try to create, scale, or upgrade an AKS cluster, you may receive the following error message:

Unable to establish outbound connection from agents, please see <https://aka.ms/aks-required-ports-and-addresses> for more information.

Details: Code="VMExtensionProvisioningError"

Message="VM has reported a failure when processing extension 'vmssCSE'.

Error message: "Enable failed: failed to execute command: command terminated with exit status=50\n[stdout]\n[nstderr]\nncc: connect to mcr.microsoft.com port 443 (tcp) failed: Connection timed out\nCommand exited with non-zero status

Error details : "vmssCSE error messages : {vmssCSE exit status=50, output=pt/apt.conf.d/95proxy...}

This error may also cause your running nodes to become NotReady or cause image pull failures because outbound connectivity is blocked from some or all nodes in your cluster.

Cause

The custom script extension that downloads the necessary components to provision the nodes couldn't establish the necessary outbound connectivity to obtain packages. For public clusters, the nodes try to communicate with the Microsoft Container Registry (MCR) endpoint (`mcr.microsoft.com`) on port 443.

There are many reasons why the outbound traffic might be blocked. The best way to troubleshoot outbound connectivity failures is by running a connectivity analysis with [Azure Virtual Network Verifier \(Preview\)](#). By running a connectivity analysis, you can visualize the hops within the traffic flow and any misconfigurations within Azure networking resources that are blocking traffic. To manually troubleshoot outbound connectivity failures, you can use the Secure Shell protocol (SSH) to connect to the node. This section covers instructions for both types of investigations:

Check if Azure network resources are blocking traffic to the endpoint

To determine if traffic is blocked to the endpoint due to Azure network resources, run a connectivity analysis from your AKS cluster nodes to the endpoint using the [Azure Virtual Network Verifier \(Preview\)](#) tool. The connectivity analysis covers the following resources:

- Azure Load Balancer
- Azure Firewall
- A network address translation (NAT) gateway
- Network security group (NSG)
- Network policy
- User defined routes (route tables)
- Virtual network peering

Note

Azure Virtual Network Verifier (Preview) can't access any external or third-party networking resources, such as a custom firewall. If the connectivity analysis doesn't detect any blocked traffic, we recommend that you perform a manual check of any external networking to cover all hops in the traffic flow.

Currently, clusters using Azure CNI Overlay aren't supported for this feature. Support for CNI Overlay is planned for August 2025.

1. Navigate to your cluster in the Azure portal. In the sidebar, navigate to the Settings -> Node pools blade.
2. Identify the nodepool you want to run a connectivity analysis from. Click on the nodepool to select it as the scope.
3. Select "Connectivity analysis (Preview)" from the toolbar at the top of the page. If you don't see it, click on the three dots "..." in the toolbar at the top of the page to open the expanded menu.
image
4. Select a Virtual Machine Scale Set (VMSS) instance as the source. The source IP addresses are populated automatically.
5. Select a public domain name/endpoint as the destination for the analysis, one example is `mcr.microsoft.com`. The destination IP addresses are also populated automatically.
6. Run the analysis and wait up to 2 minutes for the results. In the resulting diagram, identify the associated Azure network resources and where traffic is blocked. To view the detailed analysis output, click on the "JSON output" tab or click into the arrows in the diagram.

Manual troubleshooting

If the Azure Virtual Network Verifier (Preview) tool doesn't provide enough insight into the issue, you can manually troubleshoot outbound connectivity failures by using the Secure Shell protocol (SSH) to connect to the node. To make the connection, follow the instructions in [Connect to Azure Kubernetes Service \(AKS\) cluster nodes for maintenance or troubleshooting](#).

Then, test the connectivity on the cluster by following these steps:

1. After you connect to the node, run the `nc` and `dig` commands:

Bash

```
nc -vz mcr.microsoft.com 443  
dig mcr.microsoft.com 443
```

ⓘ Note

If you can't access the node through SSH, you can test the outbound connectivity by running the [`az vmss run-command invoke`](#) command against the Virtual Machine Scale Set instance:

Azure CLI

```
# Get the VMSS instance IDs.  
az vmss list-instances --resource-group <mcr-resource-group-name> \  
--name <vmss-name> \  
--output table
```

```
# Use an instance ID to test outbound connectivity.
az vmss run-command invoke --resource-group <mc-resource-group-name> \
    --name <vmss-name> \
    --command-id RunShellScript \
    --instance-id <vmss-instance-id> \
    --output json \
    --scripts "nc -vz mcr.microsoft.com 443"
```

2. If you try to create an AKS cluster by using an HTTP proxy, run the `nc`, `curl`, and `dig` commands after you connect to the node:

Bash

```
# Test connectivity to the HTTP proxy server from the AKS node.
nc -vz <http-s-proxy-address> <port>

# Test traffic from the HTTP proxy server to HTTPS.
curl --proxy http://<http-proxy-address>:<port>/ --head
https://mcr.microsoft.com

# Test traffic from the HTTPS proxy server to HTTPS.
curl --proxy https://<https-proxy-address>:<port>/ --head
https://mcr.microsoft.com

# Test DNS functionality.
dig mcr.microsoft.com 443
```

① Note

If you can't access the node through SSH, you can test the outbound connectivity by running the `az vmss run-command invoke` command against the Virtual Machine Scale Set instance:

Azure CLI

```
# Get the VMSS instance IDs.
az vmss list-instances --resource-group <mc-resource-group-name> \
    --name <vmss-name> \
    --output table

# Use an instance ID to test connectivity from the HTTP proxy server to
# HTTPS.
az vmss run-command invoke --resource-group <mc-resource-group-name> \
    --name <vmss-name> \
    --command-id RunShellScript \
    --instance-id <vmss-instance-id> \
    --output json \
```

```

--scripts "curl --proxy http://<http-proxy-address>:<port>/ --head
https://mcr.microsoft.com"

# Use an instance ID to test connectivity from the HTTPS proxy server to
# HTTPS.
az vmss run-command invoke --resource-group <mc-resource-group-name> \
    --name <vmss-name> \
    --command-id RunShellScript \
    --instance-id <vmss-instance-id> \
    --output json \
    --scripts "curl --proxy https://<https-proxy-address>:<port>/ --head
https://mcr.microsoft.com"

# Use an instance ID to test DNS functionality.
az vmss run-command invoke --resource-group <mc-resource-group-name> \
    --name <vmss-name> \
    --command-id RunShellScript \
    --instance-id <vmss-instance-id> \
    --output json \
    --scripts "dig mcr.microsoft.com 443"

```

Solution

The following table lists specific reasons why traffic might be blocked, and the corresponding solution for each reason:

[] [Expand table](#)

Issue	Solution
Traffic is blocked by firewall rules, a proxy server, or Network Security Group (NSG)	This issue occurs when the AKS required ports or Fully Qualified Domain Names (FQDNs) are blocked by a firewall, proxy server, or NSG. Ensure these ports and FQDNs are allowed. To determine what's blocked, check the connectivity provided in the preceding Cause section. For more information about AKS required ports and FQDNs, see Outbound network and FQDN rules for Azure Kubernetes Service (AKS) clusters .
The AAAA (IPv6) record is blocked on the firewall	On your firewall, verify that nothing exists that would block the endpoint from resolving in Azure DNS.
Private cluster can't resolve internal Azure resources	In private clusters, the Azure DNS IP address (168.63.129.16) must be added as an upstream DNS server if custom DNS is used. Verify that the address is set on your DNS servers. For more information, see Create a private AKS cluster and What is IP address 168.63.129.16? .

More information

- General troubleshooting of AKS cluster creation issues

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

RequestDisallowedByPolicy error when deploying an AKS cluster

Article • 04/10/2025

This article discusses how to identify and resolve the `RequestDisallowedByPolicy` error that occurs when you try to create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to deploy an AKS cluster, you receive the following error message:

Resource request failed due to RequestDisallowedByPolicy. Please see [https://aka.ms/aks-](https://aka.ms/aks-requestdisallowedbypolicy)

[requestdisallowedbypolicy](#) for more details. The detailed error message:

Code="RequestDisallowedByPolicy"

Message="Resource 'MC_clustername' was disallowed by policy."

Cause

For security or compliance, your subscription administrators might assign policies that limit how resources are deployed. For example, your subscription might have a policy that prevents you from creating public IP addresses, network security groups, user-defined routes, or route tables. The error message includes the specific reason why the cluster creation was blocked.

! Note

Only you can manage the policies in your environment. Microsoft can't disable or bypass those policies.

Solution

To fix this issue, follow these steps:

1. Find the policy that blocks the action. These policies are listed in the error message.

The name of a policy assignment or definition is the last segment of the `id` string that's shown in the error message.

```
# Example
Code: RequestDisallowedByPolicy
Message: Resource 'resourcegroup' was disallowed by policy. Policy
identifiers: '[{"policyAssignment":{"name":"Not allowed resource
types","id":"/subscriptions/00000000-0000-0000-
000000000000/providers/Microsoft.Authorization/policyAssignments/000000000000
000000000000"}, "policyDefinition":{"name":"Not allowed resource
types","id":"/subscriptions/00000000-0000-0000-0000-
000000000000/providers/Microsoft.Authorization/policyDefinitions/not-allowed-
resourcetypes","version":"1.0.0"}}]'.
```

2. If possible, update your deployment to comply with the policy restrictions, and then retry the deployment. Alternatively, if you have permission to update policy, [add an exemption](#) to the policy.

To get details about the policy that blocked your cluster deployment, see [RequestDisallowedByPolicy error with Azure resource policy](#).

 **Note**

After you fix the policy that blocks the AKS cluster creation, run the `az aks update -g MyResourceGroup -n MyManagedCluster` command to change the cluster from a failed state to a successful state. This change reconciles the cluster and retries the last failed operation. For more information about clusters in a failed state, see [Troubleshoot Azure Kubernetes Service clusters or nodes in a failed state](#).

More information

[General troubleshooting of AKS cluster creation issues](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the ServicePrincipalValidationClientError error code

Article • 10/23/2024

This article discusses how to identify and resolve the `ServicePrincipalValidationClientError` error that might occur if you try to create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- [Azure CLI](#), version 2.0.59 or a later version. If Azure CLI is already installed, you can find the version number by running `az --version`.

Symptoms

When you try to deploy an AKS cluster, you receive the following error message:

```
adal: Refresh request failed. Status Code = '401'.
```

Response body: {

```
    "error": "invalid_client",
```

```
    "error_description": "AADSTS7000215: Invalid client secret provided. Ensure the secret being sent in the request is the client secret value, not the client secret ID, for a secret added to app '123456789-1234-1234-1234-1234567890987'.\r\n
```

Trace ID: 12345\r\n

Correlation ID: 6789\r\n

Timestamp: 2022-02-03 03:07:11Z",

"error_codes": [7000215],

"timestamp": "2022-02-03 03:07:11Z",

"trace_id": "12345",

"correlation_id": "6789",

```
"error_uri": "https://login.microsoftonline.com/error?code=7000215" }

} Endpoint https://login.microsoftonline.com/123456787/oauth2/token?api-version=1.0
```

Cause

The secret that's provided for the highlighted service principal isn't valid.

Solution 1: Reset the service principal secret

To resolve this issue, reset the service principal secret by using one of the following methods:

- Reset the service principal's credential by running the [az ad sp credential reset](#) command:

Azure CLI

```
az ad sp credential reset --name "01234567-89ab-cdef-0123-456789abcdef"
--query password --output tsv
```

- Specify the expiration date by running the following command:

Azure CLI

```
az ad sp credential reset --name <service-principal-name> --credential-description "New secret for AKS" --years 1
```

The preceding command resets the secret and displays it as output. Then, you can specify the new secret when you try to create the new cluster again.

For failed operations in an existing cluster, ensure that you update your AKS cluster with the new secret:

Azure CLI

```
az aks update-credentials --resource-group <resource-group> --name <aks-cluster> --reset-service-principal --client-secret <new-client-secret>
```

Solution 2: Create a new service principal

You can create a new service principal and get the secret that's associated with it by running the `az ad sp create-for-rbac` command:

```
Azure CLI
```

```
az ad sp create-for-rbac --role Contributor
```

The output of the command should resemble the following JSON string:

```
JSON
```

```
{  
  "appId": "12345678-9abc-def0-1234-56789abcdef0",  
  "name": "23456789-abcd-ef01-2345-6789abcdef01",  
  "password": "3456789a-bcde-f012-3456-789abcdef012",  
  "tenant": "456789ab-cdef-0123-4567-89abcdef0123"  
}
```

Note the `appId` and `password` values that are generated. After you get these values, you can rerun the cluster creation command for the new service principal and secret.

To update your AKS cluster with the new service principal's credential, run the following command:

```
Azure CLI
```

```
az aks update-credentials --resource-group <resource-group> --name <aks-cluster> --service-principal <new-client-id> --client-secret <new-client-secret>
```

More information

- General troubleshooting of AKS cluster creation issues

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Provide product feedback 

Troubleshoot the SubscriptionRequestsThrottled error code (429)

Article • 09/10/2024

This article discusses how to identify and resolve the `SubscriptionRequestsThrottled` error (status 429) that occurs when you try to create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to create an AKS cluster, you receive the following "Reconcile standard load balancer failed" error message that shows a "SubCode" value of `SubscriptionRequestsThrottled` and a "Status" value of `429`:

Reconcile standard load balancer failed.

Details: outboundReconciler retry failed:

Category: ClientError;

SubCode: `SubscriptionRequestsThrottled`;

Dependency: Microsoft.Network/PublicIPAddresses;

OrginalError: autorest/azure: Service returned an error. `Status=429`

`Code="SubscriptionRequestsThrottled"`

Message="Number of requests for subscription '`<subscription-id-guid>`' and operation

'GET/SUBSCRIPTIONS/RESOURCEGROUPS/PROVIDERS/MICROSOFT.NETWORK/PUBLICIPADDRESSES' exceeded the backend storage limit. Please try again after '6' seconds.";

AKSTeam: Networking, Retriable: false.

Request throttling can occur on different Azure components, so the error message might be different based on the kind of resource this issue is occurring on.

Cause

Azure Resource Manager requests are being throttled. For information about how Azure Resource Manager limits work, and the specific limits per hour, see [Throttling Resource Manager requests](#).

Solution 1: Use another subscription

If you have access to a different subscription, you can simply deploy the cluster to that subscription.

Solution 2: Modify your access patterns

To resolve this issue, examine your access patterns for the throttled subscription. The following table lists the possible access patterns and corresponding solutions.

[] [Expand table](#)

Access pattern	Solution
Automated scripts constantly scan the subscription	Run the scripts less frequently
Many users access the subscription	Have each user use their own subscription
Scripts scan every storage account in the subscription	Scope the script to query only the resources that it must have

More information

- [General troubleshooting of AKS cluster creation issues](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Troubleshoot the Throttled error code (429)

Article • 03/06/2025

This article discusses how to identify and resolve the **Throttled** error (status 429) that occurs when you try to create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to create an AKS cluster, you receive the following "The PutManagedClusterHandler.PUT request limit has been exceeded" error message that shows a "SubCode" value of **Throttled** and a "Status" value of **429**:

Category: ClientError;

SubCode: Throttled;

OrginalError: autorest/azure: Service returned an error. **Status=429**

Code="Throttled"

Message="> The PutManagedClusterHandler.PUT request limit has been exceeded for SubID='<*subscription-id-guid*>', please retry again in X seconds. For more information, please visit aka.ms/aks/throttling"; Request throttling can occur on various Azure components, so the error message might be different depending on the type of resource where this issue occurs.

Resource provider throttling is independent of ARM throttling and is tailored to the operations of a specific resource provider. In this scenario, AKS resource provider throttling is specific to the AKS resource provider and applies only to operations related to AKS resources.

Cause

AKS requests are throttled. For information about how AKS limits work and the specific limits per hour, see [Throttling limits on AKS resource provider APIs](#).

Solution

To resolve this issue, examine and modify your access pattern of the throttled subscription. The following table lists the possible access patterns and corresponding solutions.

[+] Expand table

Access pattern	Solution
Automated scripts constantly run LIST operations against managedCluster resources.	Run the scripts less frequently.
Users attempt to deploy multiple AKS clusters in a short period of time.	Space out deployments or use different subscriptions.
Users attempt to modify the same AKS cluster multiple times consecutively.	Space out operations. Ensure successful completion before initiating another one.
Users attempt to add, modify, or delete one or more agentPools on the same AKS cluster.	Space out operations. Ensure successful completion before initiating another one.

More information

[General troubleshooting of AKS cluster creation issues](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

Troubleshoot the AksCapacityHeavyUsage error code

06/27/2025

This article discusses how to identify and resolve the `AksCapacityHeavyUsage` error that might occur when you create a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to create an AKS cluster, you receive the following error message:

Code: AksCapacityHeavyUsage

Message: AKS is experiencing heavy usage in region <Region>. We are working on adding new capacity. In the meantime, please consider creating new AKS clusters in a different region. For a list of all the Azure regions, visit <https://aka.ms/aks/regions>. For more details on this error, visit <https://aka.ms/akscapacityheavyusage>.

Cause

You're trying to create a cluster in a region that has limited capacity.

When you create an AKS cluster, Microsoft Azure allocates compute resources to your subscription. You might occasionally experience the `AksCapacityHeavyUsage` error because of significant growth in demand for Azure Kubernetes Service in specific regions.

Resolution

Solution 1: Select a different region

The easiest and quickest solution is to try to deploy to a different region (for example, NorthEurope instead of WestEurope or UAENorth instead of QatarCentral). To find nearby regions, visit the [Azure Geographies page](#).

This approach might not be feasible if you already have existing resources in the requested region, but it's the preferred solution in a dev/test scenario.

Solution 2: Deploy a cluster that has different settings

The infrastructure that hosts AKS-managed clusters have different allocation reservations. Therefore, AKS might have more capacity for public clusters than it has for private clusters. If you experience the `AksCapacityHeavyUsage` error when you try to create a private cluster, try to create a public cluster instead (or vice versa).

Solution 3: Use an Azure Enterprise subscription

When capacity is running low, non-Enterprise Agreement (EA) subscriptions are limited first in AKS cluster creation to reserve resources for real production scenarios. If you have an EA subscription, make sure that you use the EA subscription to create the AKS cluster.

Solution 4: Retry the operation

Capacity is often reclaimed when other users stop or delete their AKS clusters. Therefore, the operation might succeed if you retry it later.

More information

- Ensuring capacity for users is a top priority for Microsoft, and we're working to scale up our infrastructure to accommodate the increasing popularity of Azure services.

For more information about improvements that we're making toward delivering a resilient cloud supply chain, see [this September 2021 Azure Blog article](#).

- [General troubleshooting of AKS cluster creation issues](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the VMExtensionProvisioningTimeout error code

Article • 09/23/2024

This article discusses how to identify and resolve the `VMExtensionProvisioningTimeout` error that occurs when you try to create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- [Azure CLI](#), version 2.28.0 or a later version. If Azure CLI is already installed, you can find the version number by running `az --version`.

Symptoms

When you try to create an AKS cluster by using the Azure CLI, you receive the "VMExtensionProvisioningTimeout" error with text like the following example:

```
Output

Failed to reconcile agent pool agentpool0: err: VMSSAgentPoolReconciler
retry failed:
Category: InternalError;
SubCode: VMExtensionProvisioningTimeout;
Dependency: Microsoft.Compute/VirtualMachineScaleSet;
OrginalError:
Code="VMExtensionProvisioningTimeout"
Message="Provisioning of VM extension vmssCSE has timed out. Extension
provisioning has taken too long to complete. The extension last reported
\"Plugin enabled\".\r\n\r\nMore information on troubleshooting is available
at <https://aka.ms/VMExtensionCSELinuxTroubleshoot>";
AKSTeam: NodeProvisioning,
Retriable: true
```

You also can [view the error details in the Azure portal](#).

Cause

Several different issues can cause the "VMExtensionProvisioningError" class of errors. However, the troubleshooting steps are the same for all the issues. Possible causes are as follows:

- The custom script extension that provisions the virtual machines (VMs) can't establish a connection to the endpoint that's used for downloading the Kubernetes binaries.
- The custom script extension that provisions the VMs can't establish a connection to the endpoint that's used for downloading the CNI binaries.
- The custom script extension that provisions the VMs can't establish the required outbound connectivity to obtain packages.
- The cluster can't resolve the necessary Domain Name System (DNS) address to correctly provision the node.
- The custom script extension that provisions the VMs reached a timeout while running a packet management update (such as [apt-get ↗](#) in case the node pool uses Linux).

Solution

Follow these steps:

1. If egress filtering is set up on the cluster (such as [custom user-defined routes](#)), see [Limit network traffic with Azure Firewall in Azure Kubernetes Service \(AKS\)](#) and [Outbound network and FQDN rules for AKS clusters](#) to view the necessary prerequisites, and make sure that your setup meets the prerequisites.
2. On your DNS servers and firewall, make sure that nothing blocks the resolution of your cluster's fully qualified domain name (FQDN).
3. Because your custom DNS server might be configured incorrectly, review the following articles if FQDN resolution continues to be blocked:
 - [Create a private AKS cluster](#)
 - [Private Azure Kubernetes service with custom DNS server ↗](#)
 - [What is IP address 168.63.129.16?](#)

More information

- [General troubleshooting of AKS cluster creation issues](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Troubleshoot WINDOWS_CSE_ERROR_CHECK_API_SERVER_CONNECTIVITY error (5)

Article • 04/14/2025

This article discusses how to identify and resolve the "WINDOWS_CSE_ERROR_CHECK_API_SERVER_CONNECTIVITY" error (5) that occurs when you try to add a Windows node pool in an Azure Kubernetes Service (AKS) cluster.

Prerequisites

One of the following tools is required:

- The PowerShell command-line shell for Windows nodes.
- The [Netcat ↗](#) (nc) command-line tool for Linux nodes.

Symptoms

When you try to add a Windows node pool in an AKS cluster, you receive the following error message:

```
Code="VMExtensionProvisioningError"
Message="CSE Error: WINDOWS_CSE_ERROR_CHECK_API_SERVER_CONNECTIVITY." Exit
Code: 5. Details: Unable to establish connection from agents to Kubernetes API server.
```

Cause

Your cluster nodes can't connect to the cluster API server pod.

Troubleshooting steps

1. Connect to the respective node by following the steps described in [Windows Server proxy connection for SSH](#):
2. Verify that your nodes can resolve the cluster's fully qualified domain name (FQDN):

On existing Windows nodes, run the following command:

```
PowerShell
```

```
Test-NetConnection -ComputerName <cluster-fqdn> -Port 443
```

Or, on existing Linux nodes, run the following command:

Bash

```
nc -vz <cluster-fqdn> 443
```

3. If the command output shows `False` or `Timeout`, check your network configuration. For example, check whether you set "Deny" rules for the API server in network security groups (NSGs) of the virtual network.
4. If you're using egress filtering through a firewall, make sure that traffic is allowed to your cluster FQDN.
5. If you've authorized IP addresses that are enabled on your cluster, the firewall's outbound IP address can be blocked. In this scenario, you must add the outbound IP address of the firewall to the list of authorized IP ranges for the cluster. For more information, see [Secure access to the API server using authorized IP address ranges in AKS](#).

References

- [General troubleshooting of AKS cluster creation issues](#)
- [Exit codes in Windows CSE](#)

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the ZonalAllocationFailed, AllocationFailed, or OverconstrainedAllocationRequest error code

Article • 09/05/2024

This article describes how to identify and resolve the `ZonalAllocationFailed`, `AllocationFailed`, or `OverconstrainedAllocationRequest` error that might occur when you try to create, deploy, or update a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- [Azure CLI](#) (optional), version 2.0.59 or a later version. If Azure CLI is already installed, you can find the version number by using `az --version`.
- [Azure PowerShell](#) (optional).

Symptoms

When you try to create an AKS cluster, you receive the following error message:

Reconcile vmss agent pool error: VMSSAgentPoolReconciler retry failed:

Category: InternalError;

SubCode: ZonalAllocationFailed;

Dependency: Microsoft.Compute/VirtualMachineScaleSet;

OrginalError: Code="ZonalAllocationFailed"

Message="Allocation failed. We do not have sufficient capacity for the requested VM size in this zone. Read more about improving likelihood of allocation success at <https://aka.ms/allocation-guidance>";

AKSTeam: NodeProvisioning

Or, when you try to upgrade or scale up a cluster, you receive the following error message:

`Code="OverconstrainedAllocationRequest"`

Message="Allocation failed. VM(s) with the following constraints cannot be allocated, because the condition is too restrictive. Please remove some constraints and try again."

Or, when you use dedicated hosts in a cluster and try to create or scale up a node pool, you receive the following error message:

`Code="AllocationFailed"`

Message="Allocation failed. VM allocation to the dedicated host failed. Please ensure that the dedicated host has enough capacity or try allocating elsewhere."

Cause 1: Limited zone availability in a SKU

You're trying to deploy, upgrade or scale up a cluster in a zone that has limited availability for the specific SKU.

Solution 1: Use a different SKU, zone, or region

Try one or more of the following methods:

- Redeploy the cluster in the same region by using a different SKU.
- Redeploy the cluster in a different zone in that region.
- Redeploy the cluster in a different region.
- Create a new node pool in a different zone or use a different SKU.

For more information about how to fix this error, see [Resolve errors for SKU not available](#).

Cause 2: Too many constraints for a virtual machine to accommodate

If you receive an `OverconstrainedAllocationRequest` error code, the Azure Compute platform can't allocate a new virtual machine (VM) to accommodate the required constraints. These constraints usually (but not always) include the following items:

- VM size
- VM SKU
- Accelerated networking

- Availability zone
- Ephemeral disk
- Proximity placement group (PPG)

Solution 2: Don't associate a proximity placement group with the node pool

If you receive an `OverconstrainedAllocationRequest` error code, you can try to create a new node pool that isn't associated with a proximity placement group.

Cause 3: Not enough dedicated hosts or fault domains

You're trying to deploy a node pool in a dedicated host group that has limited capacity or doesn't satisfy the fault domain constraint.

Solution 3: Ensure you have enough dedicated hosts for your AKS nodes/VMSS

As per [Planning for ADH Capacity on AKS](#), you're responsible for planning enough dedicated hosts to span as many fault domains as required by your AKS VMSS. For example, if the AKS VMSS is created with `FaultDomainCount=2`, you need at least two dedicated hosts in different fault domains (`FaultDomain 0` and `FaultDomain 1`).

More information

Ensuring capacity for users is a top priority for Microsoft, and we're working around the clock to reach this goal. The increasing popularity of Azure services emphasizes the need to scale up our infrastructure even more rapidly. With that in mind, we're expediting expansions and improving our resource deployment process to respond to strong customer demand. We're also adding a large amount of computing infrastructure monthly.

We have identified several methods to improve how we load-balance under a high-resource-usage situation and how to trigger the timely deployment of needed resources. Additionally, we're significantly increasing our capacity and will continue to plan for strong demand across all regions. For more information about the

improvements that we're making toward delivering a resilient cloud supply chain, see [Advancing reliability through a resilient cloud supply chain](#).

References

- General troubleshooting of AKS cluster creation issues
- Virtual Machine Scale Sets - What to expect when using proximity placement groups
- Fix an AllocationFailed or ZonalAllocationFailed error when you create, restart, or resize Virtual Machine Scale Sets in Azure

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?



"QuotaExceeded" or "InsufficientVCPUQuota" error during AKS creation or upgrade

Article • 03/07/2025

This article provides a solution to a "QuotaExceeded" or "InsufficientVCPUQuota" error that occurs during an Azure Kubernetes Service (AKS) creation or upgrade operation.

Symptoms

When you try to scale up your node pools in an AKS cluster, deploy a new cluster, or deploy a new node pool, you receive an error message that resembles the following text:

- Family quota errors

Code: QuotaExceeded

Message: Operation could not be completed as it results in exceeding approved standardDSv3Family Cores quota. Additional details - Deployment Model: Resource Manager, Location: eastus2, Current Limit: 1500, Current Usage: 1500, Additional Required: 16, (Minimum) New Limit Required: 1516.

Or

Code: ErrCode_InsufficientVCPUQuota

Message: Insufficient vcpu quota requested 48, remaining 32 for family standardDSv5Family for region qatarcentral

- Regional quota errors

Code: QuotaExceeded

Message: Operation could not be completed as it results in exceeding approved Total Regional Cores quota. Additional details - Deployment Model: Resource Manager, Location: eastus, Current Limit: 350, Current Usage: 348, Additional Required: 4, (Minimum) New Limit Required: 352.

Or

Code: ErrCode_InsufficientVCPUQuota

Message: "Insufficient regional vcpu quota left for location eastus. left regional vcpu quota 0, requested quota 4"

Cause

This error occurs because the operation that you try to perform exceeds the current quota limits.

Solution

To resolve family quota errors, [increase VM-family vCPU quotas](#).

To resolve regional quota errors, select a different region or [increase a regional vCPU quota](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

AADSTS7000222 - BadRequest or InvalidClientSecret error

07/09/2025

This article discusses how to identify and resolve the `AADSTS7000222` error (`BadRequest` or `InvalidClientSecret`) that occurs when you try to create or upgrade a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- [Azure CLI](#)

Symptoms

When you try to create or upgrade an AKS cluster, you receive one of the following error messages.

[Expand table](#)

Error code	Message
<code>BadRequest</code>	The credentials in <code>ServicePrincipalProfile</code> were invalid. Please see https://aka.ms/aks-sp-help for more details. (Details: adal: Refresh request failed. Status Code = '401'. Response body: {"error": "invalid_client", "error_description": "AADSTS7000222: The provided client secret keys for app '<application-id>' are expired. Visit the Azure portal to create new keys for your app: https://aka.ms/NewClientSecret , or consider using certificate credentials for added security: https://aka.ms/certCreds ."}.
<code>InvalidClientSecret</code>	Customer auth is not valid for tenant: <tenant-id>: adal: Refresh request failed. Status Code = '401'. Response body: {"error": "invalid_client", "error_description": "AADSTS7000222: The provided client secret keys for app '<application-id>' are expired. Visit the Azure portal to create new keys for your app: https://aka.ms/NewClientSecret , or consider using certificate credentials for added security: https://aka.ms/certCreds ."}.

Cause

The issue that generates this service principal alert usually occurs for one of the following reasons:

- The client secret expired.
- Incorrect credentials were provided.
- The service principal doesn't exist within the Microsoft Entra ID tenant of the subscription.

Verify the cause

Use the following commands to retrieve the service principal profile for your AKS cluster and check the expiration date of the service principal. Make sure to set the appropriate variables for your AKS resource group and cluster name.

Azure CLI

```
SP_ID=$(az aks show --resource-group $RESOURCE_GROUP_NAME \
--name $AKS_CLUSTER_NAME \
--query servicePrincipalProfile.clientId \
--output tsv)
az ad app credential list --id "$SP_ID"
```

Alternatively, you can verify that the service principal name and secret are correct and aren't expired. To do this, follow these steps:

1. In the [Azure portal](#), search for and select **Microsoft Entra ID**.
2. In the navigation pane of Microsoft Entra ID, select **App registrations**.
3. On the **Owned applications** tab, select the affected application.
4. Find the service principal name and secret information, and verify that the information is correct and current.

Solution

1. In the [Update or rotate the credentials for an AKS cluster](#) article, follow the instructions in one of the following article sections, as appropriate:
 - [Reset the existing service principal credentials](#)
 - [Create a new service principal](#)
2. Using your new service principal credentials, follow the instructions in the [Update AKS cluster with service principal credentials](#) section of that article.

More information

- Use a service principal with Azure Kubernetes Service (AKS) (especially the Troubleshoot section)

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the DnsServiceIpOutOfServiceCidr error

Article • 10/25/2024

This article discusses how to identify and resolve the `DnsServiceIpOutOfServiceCidr` error that occurs when you try to create or upgrade an Azure Kubernetes Service (AKS) cluster.

Prerequisites

This article requires Azure CLI version 2.0.65 or a later version. To find the version number, run `az --version`. If you have to install or upgrade Azure CLI, see [How to install the Azure CLI](#).

For more detailed information about the upgrade process, see [Upgrade an AKS cluster](#).

Symptoms

An AKS cluster create operation fails, and you receive a `DnsServiceIpOutOfServiceCidr` error message.

```
(DnsServiceIpOutOfServiceCidr) The DNS service IP 10.0.0.10 is out of the range defined by service CIDR 10.200.0.0/16.
```

- **Code:** `DnsServiceIpOutOfServiceCidr`
- **Message:** The DNS service IP 10.0.0.10 is out of the range defined by service CIDR 10.200.0.0/16.
- **Target:** networkProfile.dnsServiceIP

Cause

This error occurs if the DNS service IP (`--dns-service-ip`) for AKS is out of the range that's defined by the service CIDR (`--service-cidr`).

Solution

Make sure that the DNS service IP is within the range that's defined by service CIDR.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

Can't upgrade AKS cluster because of NodePoolMcVersionIncompatible error

Article • 10/23/2024

This article discusses how to resolve the "NodePoolMcVersionIncompatible - Node pool version 1.x.y and control plane version 1.a.b are incompatible" error that occurs when you upgrade a node pool in a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- [Azure CLI](#)

Symptoms

When you upgrade a node pool in an AKS cluster, you receive one of the following error messages:

BadRequest - NodePoolMcVersionIncompatible

Error: Node pool version 1.x.y and control plane version 1.a.b are incompatible. Minor version of node pool cannot be more than 2 versions less than control plane's version. Minor version of node pool is x and control plane is a. For more information, please check <https://aka.ms/version-skew-policy>.

Resource ID:

/subscriptions/<subscription_id>/resourcegroups/<aks_cluster_resource_group>/providers/Microsoft.ContainerService/managedClusters/<aks_cluster_name>.

BadRequest - NodePoolMcVersionIncompatible

Error: Node pool version 1.x.y and control plane version 1.a.b are incompatible. Minor version of node pool version x is bigger than control plane version a. For more information, please check <https://aka.ms/version-skew-policy>.

Resource ID:

/subscriptions/<subscription_id>/resourcegroups/<aks_cluster_resource_group>/providers/Microsoft.ContainerService/managedClusters/<aks_cluster_name>.

Cause

These issues occur if you try to upgrade a node pool that's more than two versions behind the AKS control plane version, or if you try to add a node pool that's at a more recent version than the control plane version.

You must meet the following conditions when you upgrade a node pool:

- The node pool version can't be greater than the control `<major>.<minor>. <patch>` version.
- The node pool version must be within two *minor* versions of the control plane version.

For more information, see the [AKS validation rules for upgrade](#).

Solution 1: Make sure that the node pool version is within two minor versions of the control plane version

1. [Get the control plane version](#) by running the `az aks get-upgrades` command in Azure CLI.

Here's an example use of the command. The `MasterVersion` output column contains the control plane version.

Azure CLI			
<code>az aks get-upgrades --resource-group aksrg --name testcluster1 --output table</code>			
Output			
Name	ResourceGroup	MasterVersion	Upgrades
default	aksrg	1.23.12	1.23.15, 1.24.6, 1.24.9

2. [Upgrade the node pool](#) by running the `az aks nodepool upgrade` Azure CLI command, and provide a Kubernetes version that's within two minor versions of the control plane version.

For example, if the control plane version is `1.23.12`, you can specify the Kubernetes version of the node pool as `1.23.8` or `1.23.12`.

Here's an example use of the command:

Azure CLI

```
az aks nodepool upgrade \
--resource-group aksrg \
--cluster-name testcluster1 \
--name mynodepool \
--kubernetes-version 1.23.8 \
--no-wait
```

Solution 2: Make sure that the node pool version isn't greater than the control plane version

1. [Get the control plane version](#) by running the `az aks get-upgrades` command in Azure CLI.

Here's an example use of the command. The `MasterVersion` output column contains the control plane version.

Azure CLI

```
az aks get-upgrades --resource-group aksrg --name testcluster1 --output
table
```

Output

Name	ResourceGroup	MasterVersion	Upgrades
default	aksrg	1.23.12	1.23.15, 1.24.6, 1.24.9

2. [Upgrade the node pool](#) by running the `az aks nodepool upgrade` Azure CLI command, and provide a Kubernetes version that's less than or equal to the control plane version.

Here's an example use of the command:

Azure CLI

```
az aks nodepool upgrade \
--resource-group aksrg \
--cluster-name testcluster1 \
--name mynodepool \
```

```
--kubernetes-version 1.23.12 \
--no-wait
```

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

Troubleshoot the "PublicIPCountLimitReached" error code

Article • 10/16/2024

This article explains how to identify and resolve the "PublicIPCountLimitReached" error. This error occurs when you try to create, update, or upgrade an Azure Kubernetes Service (AKS) cluster. This error can also be caused by any other action that triggers the creation of a Public IP address, such as deploying a Kubernetes Service-type Public Load Balancer.

Symptoms

An AKS cluster creation, update, upgrade, or other operation that triggers the creation of a Public IP address, such as deploying a Kubernetes Service that has a Public Load Balancer, fails and returns a "PublicIPCountLimitReached" error message.

Cause

This error occurs if you've reached the maximum number of public IP addresses that are allowed for your subscription. The limit varies based on your subscription type. For more information about public IP address limits, refer to [this detailed guide](#).

Solution

To increase the public IP limit or quota for your subscription, follow these steps:

1. Navigate to the [Azure portal](#), and select the subscriptions for which you're performing the operation.
2. In the **Settings** section, select **Usage + quotas**. Set the **Provider** to **Networking**, and optionally filter by the region of your AKS cluster.
3. Locate the **Public IP Addresses** record. On the same line, select the **Create a new support request** button.

The screenshot shows the Azure portal interface for managing usage and quotas. The left sidebar is expanded to show various options under 'Usage + quotas'. The main area displays a table of quota usage for different resources in the 'East US' region. The 'Public IP Addresses' row is highlighted with a red circle labeled '2', indicating it's the target for a quota increase. The top navigation bar includes links for 'New Quota Request', 'Refresh', and 'Download'.

- On the New support request page, specify the new limit that you require, and then follow the instructions to create the support request.

The screenshot shows the 'New support request' wizard. The third step, 'Additional details', is active. It includes sections for 'Problem description', 'Provide details for the request', 'File upload', 'Advanced diagnostic information', and 'Support method'. The 'New Limit' field is highlighted with a red box. The right side of the screen shows a 'Quota details' panel with configuration for a 'Public IP Addresses' quota.

After the quota change takes effect, retry the operation that initially triggered the "PublicIPCountLimitReached" error.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Troubleshoot the "Quotaexceeded" error code

Article • 09/16/2024

This article discusses how to identify and resolve the "QuotaExceeded" error that occurs when you try to upgrade an Azure Kubernetes Service (AKS) cluster.

Here's an example of the error message:

```
Operation results in exceeding quota limits of Core. Maximum allowed: X, Current in use: X, Additional requested: X
```

Prerequisites

This article requires Azure CLI version 2.0.65 or a later version. To find the version number, run `az --version`. If you have to install or upgrade Azure CLI, see [How to install the Azure CLI](#).

For more detailed information about the upgrade process, see the "Upgrade an AKS cluster" section in [Upgrade an Azure Kubernetes Service \(AKS\) cluster](#).

Symptoms

An AKS cluster upgrade fails, and you receive a "QuotaExceeded" error message.

Cause

The issue occurs if the quota is reached. Your subscription doesn't have available resources that are required for upgrading.

Solution

To raise the limit or quota for your subscription, go to the [Azure portal](#) and file a [Service and subscription limits \(quotas\)](#) support ticket. In this case, you have to submit a support ticket to increase the quota for compute cores.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Troubleshoot the "ServiceCidrOverlapExistingSubnetsCidr" error during an AKS cluster upgrade

Article • 04/10/2024

This article discusses how to identify and resolve the "ServiceCidrOverlapExistingSubnetsCidr" error that might occur when you try to upgrade a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

An AKS cluster upgrade operation fails and displays the following error message:

```
(ServiceCidrOverlapExistingSubnetsCidr) The specified service CIDR <service-cidr-1>  
is conflicted with an existing subnet CIDR <subnet-cidr-2>  
Code: ServiceCidrOverlapExistingSubnetsCidr  
Message: The specified service CIDR <service-cidr-1> is conflicted with an existing  
subnet CIDR <subnet-cidr-2>  
Target: networkProfile.serviceCIDR
```

Cause

The service address range of a cluster is the set of virtual IP addresses that Kubernetes assigns to internal services in the cluster. This range is defined upon initial cluster creation. The range shouldn't overlap with the cluster's virtual network or any other network that can be routed from the cluster. For more information, see [Deployment parameters](#).

Before AKS starts an upgrade operation, it checks the cluster's virtual network for any existing subnet Classless Inter-Domain Routing (CIDR) address spaces that overlap with the cluster's service CIDR. If any such subnet overlap is found, the operation generates the "ServiceCidrOverlapExistingSubnetsCidr" error.

To resolve this issue, use one of the following solutions.

Solution 1: Remove the overlapping subnet

 **Note**

Use this solution if no resources are attached to the subnet.

1. Delete the subnet. To do this, follow the steps that are described in [Delete a subnet](#).
2. Retry the AKS cluster upgrade operation.

Solution 2: Adjust the overlapping subnet address range

 **Note**

Use this solution if it's acceptable to change the subnet's address range.

1. Change the subnet's address range. To do this, follow the steps that are described in [Change subnet settings](#).
2. Retry the AKS cluster upgrade operation.

Solution 3: Redeploy the cluster with a different service CIDR

 **Note**

Use this solution if it's not acceptable or not possible to remove the overlapping subnet or adjust its configuration. You can't change the cluster's service CIDR after cluster creation.

Review the [deployment parameters](#) and redeploy your cluster by using a different service CIDR.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Troubleshoot the "SubnetIsFull" error code during an AKS cluster upgrade

Article • 10/10/2024

This article discusses how to identify and resolve the "SubnetIsFull" error that occurs when you try to upgrade an Azure Kubernetes Service (AKS) cluster.

Here's an example of the error message:

```
Failed to scale node pool <AGENT POOL NAME>' in Kubernetes service '<NAME>'.
Error: VMSSAgentPoolReconciler retry failed: Code='SubnetIsFull'
Message='<SUBNET NAME>\ with address prefix <PREFIX>\ doesn't have enough
capacity for IP addresses.' Details=[]
```

Prerequisites

This article requires Azure CLI version 2.0.65 or a later version. To find the version number, run `az --version`. If you have to install or upgrade Azure CLI, see [How to install the Azure CLI](#).

For more detailed information about the upgrade process, see the "Upgrade an AKS cluster" section in [Upgrade an Azure Kubernetes Service \(AKS\) cluster](#).

Symptoms

An AKS cluster upgrade fails, and you receive a "SubnetIsFull" error message.

Cause

This error occurs if your cluster doesn't have enough IP addresses to create a new node.

When you plan to do an upgrade or scaling operation, consider the number of required IP addresses. If the IP address range that you configured in the cluster supports only a fixed number of nodes, the upgrade or scaling operation will fail. For more information, see [IP address planning for your Azure Kubernetes Service \(AKS\) clusters](#).

Solution

Reduce the cluster nodes to reserve IP addresses for the upgrade.

If scaling down isn't an option, and your virtual network CIDR has enough IP addresses, try to add a node pool that has a [unique subnet](#):

1. Add a new user node pool in the virtual network on a larger subnet.
2. Switch the original node pool to a system node pool type.
3. Scale up the user node pool.
4. Scale down the original node pool.

More information

- [InsufficientSubnetSize error code](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?



Troubleshoot UpgradeFailed errors due to eviction failures caused by PDBs

Article • 03/06/2025

This article discusses how to identify and resolve UpgradeFailed errors due to eviction failures caused by Pod Disruption Budgets (PDBs) that occur when you try to upgrade an Azure Kubernetes Service (AKS) cluster.

Prerequisites

This article requires Azure CLI version 2.67.0 or a later version. To find the version number, run `az --version`. If you have to install or upgrade Azure CLI, see [How to install the Azure CLI](#).

For more detailed information about the upgrade process, see the "Upgrade an AKS cluster" section in [Upgrade an Azure Kubernetes Service \(AKS\) cluster](#).

Symptoms

An AKS cluster upgrade operation fails with one of the following error messages:

- (UpgradeFailed) Drain node `aks-<nodepool-name>-xxxxxxxxx-vmssxxxxxx` failed when evicting pod `<pod-name>` failed with Too Many Requests error. This is often caused by a restrictive Pod Disruption Budget (PDB) policy. See <https://aka.ms/aks/debugdrainfailures>. Original error: Cannot evict pod as it would violate the pod's disruption budget.. PDB debug info: `<namespace>/<pod-name>` blocked by pdb `<pdb-name>` with 0 unready pods.
- Code: UpgradeFailed
Message: Drain node `aks-<nodepool-name>-xxxxxxxxx-vmssxxxxxx` failed when evicting pod `<pod-name>` failed with Too Many Requests error. This is often caused by a restrictive Pod Disruption Budget (PDB) policy. See <https://aka.ms/aks/debugdrainfailures>. Original error: Cannot evict pod as it would violate the pod's disruption budget.. PDB debug info: `<namespace>/<pod-name>` blocked by pdb `<pdb-name>` with 0 unready pods.

Cause

This error might occur if a pod is protected by the Pod Disruption Budget (PDB) policy. In this situation, the pod resists being drained, and after several attempts, the upgrade operation fails, and the cluster/node pool falls into a `Failed` state.

Check the PDB configuration: `ALLOWED DISRUPTIONS` value. The value should be `1` or greater. For more information, see [Plan for availability using pod disruption budgets](#). For example, you can check the workload and its PDB as follows. You should observe the `ALLOWED DISRUPTIONS` column doesn't allow any disruption. If the `ALLOWED DISRUPTIONS` value is `0`, the pods aren't evicted and node drain fails during the upgrade process:

```
Console

$ kubectl get deployments.apps nginx
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
nginx    2/2     2           2           62s

$ kubectl get pod
NAME                           READY   STATUS    RESTARTS   AGE
nginx-7854ff8877-gbr4m     1/1     Running   0          68s
nginx-7854ff8877-gnltd     1/1     Running   0          68s

$ kubectl get pdb
NAME      MIN AVAILABLE   MAX UNAVAILABLE   ALLOWED DISRUPTIONS   AGE
nginx-pdb  2               N/A                0                  24s
```

You can also check for any entries in Kubernetes events using the command `kubectl get events | grep -i drain`. A similar output shows the message "Eviction blocked by Too Many Requests (usually a pdb)":

```
Console

$ kubectl get events | grep -i drain
LAST SEEN   TYPE      REASON          OBJECT
MESSAGE
(...)
32m        Normal    Drain           node/aks-<nodepool-name>-
xxxxxxxxx-vmssxxxxxx  Draining node: aks-<nodepool-name>-xxxxxxxx-vmssxxxxxx
2m57s       Warning   Drain           node/aks-<nodepool-name>-
xxxxxxxxx-vmssxxxxxx  Eviction blocked by Too Many Requests (usually a pdb):
<pod-name>
12m        Warning   Drain           node/aks-<nodepool-name>-
xxxxxxxxx-vmssxxxxxx  Eviction blocked by Too Many Requests (usually a pdb):
<pod-name>
32m        Warning   Drain           node/aks-<nodepool-name>-
xxxxxxxxx-vmssxxxxxx  Eviction blocked by Too Many Requests (usually a pdb):
<pod-name>
32m        Warning   Drain           node/aks-<nodepool-name>-
xxxxxxxxx-vmssxxxxxx  Eviction blocked by Too Many Requests (usually a pdb):
<pod-name>
```

```
<pod-name>
31m      Warning  Drain           node/aks-<nodepool-name>-
xxxxxxxxx-vmssxxxxxx  Eviction blocked by Too Many Requests (usually a pdb):
<pod-name>
```

To resolve this issue, use one of the following solutions.

Solution 1: Enable pods to drain

1. Adjust the PDB to enable pod draining. Generally, The allowed disruption is controlled by the `Min Available / Max unavailable` or `Running pods / Replicas` parameter. You can modify the `Min Available / Max unavailable` parameter at the PDB level or increase the number of `Running pods / Replicas` to push the Allowed Disruption value to **1** or greater.
2. Try again to upgrade the AKS cluster to the same version that you tried to upgrade to previously. This process triggers a reconciliation.

Console

```
$ az aks upgrade --name <aksName> --resource-group <resourceGroupName>
Are you sure you want to perform this operation? (y/N): y
Cluster currently in failed state. Proceeding with upgrade to existing
version 1.28.3 to attempt resolution of failed cluster state.
Since control-plane-only argument is not specified, this will upgrade
the control plane AND all nodepools to version . Continue? (y/N): y
```

Solution 2: Back up, delete, and redeploy the PDB

1. Take a backup of the PDB(s) using the command `kubectl get pdb <pdb-name> -n <pdb-namespace> -o yaml > pdb-name-backup.yaml`, and then delete the PDB using the command `kubectl delete pdb <pdb-name> -n <pdb-namespace>`. After the new upgrade attempt is finished, you can redeploy the PDB just applying the backup file using the command `kubectl apply -f pdb-name-backup.yaml`.
2. Try again to upgrade the AKS cluster to the same version that you tried to upgrade to previously. This process triggers a reconciliation.

Console

```
$ az aks upgrade --name <aksName> --resource-group <resourceGroupName>
Are you sure you want to perform this operation? (y/N): y
```

```
Cluster currently in failed state. Proceeding with upgrade to existing
version 1.28.3 to attempt resolution of failed cluster state.
Since control-plane-only argument is not specified, this will upgrade
the control plane AND all nodepools to version . Continue? (y/N): y
```

Solution 3: Delete the pods that can't be drained or scale the workload down to zero (0)

1. Delete the pods that can't be drained.

ⓘ Note

If the pods are created by a Deployment or StatefulSet, they'll be controlled by a ReplicaSet. If that's the case, you might have to delete or scale the workload replicas to zero (0) of the Deployment or StatefulSet. Before you do that, we recommend that you make a backup: `kubectl get <deployment.apps -or- statefulset.apps> <name> -n <namespace> -o yaml > backup.yaml`.

2. To scale down, you can use `kubectl scale --replicas=0 <deployment.apps -or- statefulset.apps> <name> -n <namespace>` before the reconciliation
3. Try again to upgrade the AKS cluster to the same version that you tried to upgrade to previously. This process triggers a reconciliation.

Console

```
$ az aks upgrade --name <aksName> --resource-group <resourceGroupName>
Are you sure you want to perform this operation? (y/N): y
Cluster currently in failed state. Proceeding with upgrade to existing
version 1.28.3 to attempt resolution of failed cluster state.
Since control-plane-only argument is not specified, this will upgrade
the control plane AND all nodepools to version . Continue? (y/N): y
```

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Provide product feedback 

Failed to upgrade or scale Azure Kubernetes Service cluster due to missing Log Analytics workspace

Article • 09/11/2024

This article provides solutions to an "Unable to get log analytics workspace info" error that occurs when you upgrade or scale a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you start, upgrade, or scale an AKS cluster, you may see one of the following errors:

Failed to save container service '<container service name>'.

Error: Unable to get log analytics workspace info.

Resource ID:

/subscriptions/<WorkspaceSubscription>/resourcegroups/defaultresourcegroup-weu/providers/microsoft.operationalinsights/workspaces/defaultworkspace-<WorkspaceID>-weu.

Detail: operationalinsights.WorkspacesClient#GetSharedKeys: Failure responding to request: StatusCode=404 -- Original Error: autorest/azure: Service returned an error. Status=404 Code='ResourceGroupNotFound' Message='Resource group 'defaultresourcegroup-weu' could not be found.'

or

Error: Unable to retrieve log analytics workspace. If resource is intentionally deleted to stop monitoring, please also disable monitoring addon by following doc:

<https://aka.ms/aks-disable-monitoring-addon>. If resource is deleted by mistake, please re-create it by following doc: <https://aka.ms/new-log-analytics>, and enable new workspace resource id by following doc: <https://aka.ms/aks-enable-addons>.

Resource ID:

/subscriptions/<WorkspaceSubscription>/resourcegroups/defaultresourcegroup-weu/providers/microsoft.operationalinsights/workspaces/defaultworkspace-<WorkspaceID>-weu.

Detail: operationalinsights.WorkspacesClient#GetSharedKeys: Failure responding to request: StatusCode=404 -- Original Error: autorest/azure: Service returned an error.

```
Status=404 Code='ResourceGroupNotFound' Message='Resource group  
'defaultresourcegroup-weu' could not be found.'
```

This issue occurs if you delete the Log Analytics workspace or the resource group where the workspace is located without disabling monitoring on the AKS cluster.

To resolve this issue, use one of the following solutions.

Solution 1: Recover the Log Analytics workspace

If it has been less than 14 days (the default soft-delete period) since the workspace was deleted, recover the workspace.

ⓘ Note

- If it hasn't been 14 days, the workspace can't be recreated with the same name. Therefore, the recovery must be done because the AKS control plane finds the workspace based on the resource URI.
- If your workspace was deleted as part of a resource group delete operation, you must first re-create the resource group with the same name.
- To perform the workspace recovery, you must have the Contributor permissions to the subscription and resource group where the workspace is located, and the following information is also required:
 - Subscription ID
 - Resource Group name
 - Workspace name
 - Region

1. Get the workspace resource ID by running the Azure CLI command `az aks show -g <clusterRG> -n <clusterName>`.

Here's an example output of the command:

Output

```
root@AKS# az aks show -g aksrg -n testcluster1  
{ "aadProfile": null,  
  "addonProfiles": {  
    "httpapplicationrouting": {
```

```
        "config": {}, "enabled": false },
      "omsagent": {
        "config": {
          "logAnalyticsWorkspaceResourceID":
          "/subscriptions/<WorkspaceSubscription>/resourceGroups/defaultresourcegroup-eus/providers/Microsoft.OperationalInsights/workspaces/defaultworkspace-<WorkspaceID>-eus"
        },
        "enabled": true
      }
    }
  }
}
```

2. Re-create the workspace with the workspace resource ID by running the PowerShell cmdlet [New-AzOperationalInsightsWorkspace](#).
3. Run the upgrade or scale operation again.

Solution 2: Disable monitoring on the AKS cluster

If it has been more than 14 days since the workspace was deleted, disable monitoring on the AKS cluster and then run the upgrade or scale operation again.

To disable monitoring on the AKS cluster, run the following command:

Azure CLI

```
az aks disable-addons -a monitoring -g <clusterRG> -n <clusterName>
```

If the same error occurs while disabling the monitoring add-on, recreate the missing Log Analytics workspace and then run the upgrade or scale operation again.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Troubleshoot the "UnsatisfiablePDB" error during an AKS cluster upgrade

Article • 04/10/2024

This article discusses how to identify and resolve the "UnsatisfiablePDB" error that might occur when you try to [upgrade an Azure Kubernetes Service \(AKS\) cluster](#).

Prerequisites

This article requires Azure CLI version 2.53.0 or a later version. Run `az --version` to find your installed version. If you need to install or upgrade the Azure CLI, see [How to install the Azure CLI](#).

Symptoms

An AKS cluster upgrade operation fails with the following error message:

Code: UnsatisfiablePDB

Message: 1 error occurred:

* PDB < pdb-namespace >/< pdb-name > has maxunavailable == 0 can't proceed with put operation

Cause

Before starting an upgrade operation, AKS checks the cluster for any existing [Pod Disruption Budgets \(PDBs\)](#) that have the `maxUnavailable` parameter set to 0. Such PDBs are likely to block node drain operations. If node drain operations are blocked, the cluster upgrade operation can't complete successfully. This might potentially cause the cluster to be in a failed state.

After receiving the "UnsatisfiablePDB" error, you can confirm the PDB's status by running the following command:

Console

```
$ kubectl get pdb < pdb-name > -n < pdb-namespace >
```

The output of this command should be similar to the following one:

Output

NAME	MIN AVAILABLE	MAX UNAVAILABLE	ALLOWED DISRUPTIONS	AGE
< pdb-name >	N/A	0	0	49s

If the value of `MAX UNAVAILABLE` is 0, the node drain fails during the upgrade process.

To resolve this issue, use one of the following solutions.

Solution 1: Adjust the PDB's "maxUnavailable" parameter

ⓘ Note

Use this solution if you can edit the PDB resource directly.

1. Set the PDB's `maxUnavailable` parameter to `1` or a greater value. For more information, see [Specifying a PodDisruptionBudget](#).
2. Retry the AKS cluster upgrade operation.

Solution 2: Back up, delete, and redeploy the PDB

ⓘ Note

Use this solution if directly editing the PDB resource isn't viable.

1. Back up the PDB using the following command:

Console

```
$ kubectl get pdb < pdb-name > -n < pdb-namespace > -o yaml >  
pdb_backup.yaml
```

2. Delete the PDB using the following command:

Console

```
$ kubectl delete pdb < pdb-name > -n < pdb-namespace >
```

3. Retry the AKS cluster upgrade operation.
4. If the AKS cluster upgrade operation succeeds, redeploy the PDB using the following command:

Console

```
$ kubectl apply -f pdb_backup.yaml
```

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Windows Server node pools not upgraded to Gen2 during cluster node image upgrade

Article • 05/07/2025

This article discusses how to troubleshoot a scenario in which Windows Server node pools don't get upgraded automatically from Gen1 to Gen2 virtual machines (VMs) when a cluster node image is upgraded in Microsoft Azure Kubernetes Service (AKS).

ⓘ Note

This scenario doesn't apply to Gen2 VMs on Linux node pools.

Prerequisites

- Azure CLI version 2.0.81 or later. See [Install Azure CLI](#) for installation instructions.

Symptoms

Existing Windows Server node pools don't get upgraded from Gen1 to Gen2 when you [upgrade the node image](#) by using one of the following methods in Azure CLI:

- An entire cluster upgrade (by using the `az aks upgrade` command)
- A specific node pool upgrade (by using the `az aks nodepool upgrade` command)

Cause

Cause 1: Existing node pools aren't automatically upgraded to Gen2 VMs

By design, a node image upgrade doesn't support updating or upgrading existing node pools. The `az aks upgrade` and `az aks nodepool upgrade` commands upgrade only the node image (to a later node image version). These commands don't upgrade the corresponding VM generation.

Cause 2: Cluster upgrade to Kubernetes version 1.25 or a later version upgrades only the OS

When you upgrade your cluster to Kubernetes version 1.25 or a later version, only the operating system (OS) is upgraded (to Windows Server 2022). Existing node pools aren't affected. The associated Azure Virtual Machine Scale Sets contain VMs that have the same Gen1 VM.

Cause 3: Cluster upgraded and new node pool created by using Windows Server 2022 without specifying a valid VM size

After you upgrade the cluster to Kubernetes version 1.25 or a later version, you specify Windows Server 2022 as the OS to use on the node pool's VMs. However, the VMs don't use a Gen2 node image reference for one of the following reasons:

- You don't specify a VM size, and the default VM size in the region doesn't support Gen2 VMs.
- You specify a Gen1-only VM size.

When you upgrade the default OS from Windows Server 2019 (`Windows2019`) to Windows Server 2022 (`Windows2022`), the existing node pools aren't automatically upgraded to a different VM generation.

Solution

Upgrade the cluster, and then create a new Windows Server node pool that supports [Gen2 VM sizes](#) on that cluster according to the following guidelines.

 Expand table

Kubernetes cluster upgrade version	Cluster creation guidelines
1.25 or a later version	When you run the <code>az aks create</code> command to create a cluster, set the <code>--node-vm-size</code> parameter to a Gen2 VM size .
Any version earlier than 1.25	When you run the <code>az aks create</code> command to create a cluster, set the <code>--os-sku</code> parameter value to <code>Windows2022</code> , and set the <code>--node-vm-size</code> parameter value to a Gen2 VM size .

 Note

If you specify a Gen2 VM size and set the operating system as Windows Server 2019, you receive an `ErrorCode_Windows2019NotSupportedWithGen2VM` error code that's accompanied by the following error message:

<virtual-machine-size> is a Gen 2-only VM. Windows2019 does not support Gen 2 VMs. However, you can select a Gen 1 VM size or set os_sku to 2022.

To avoid this problem, choose one of the following options when you create the cluster:

- Pick a Gen1 VM size to use together with Windows Server 2019.
- Set the operating system SKU to **Windows Server 2022** to use together with your Gen2 VM.
- Before you create a new node pool, verify that the VM size supports Gen2 VMs in your region. To do this, run `az vm list-sizes --location <region> --query "[? contains(name, 'v2')].name"` --output table.
- To verify the current and available node image versions, run the following commands:
 - To check the current version: `az aks nodepool show --resource-group <resource-group> --cluster-name <cluster-name> --name <nodepool-name> --query nodeImageVersion`
 - To check the latest available version: `az aks nodepool get-upgrades --resource-group <resource-group> --cluster-name <cluster-name> --nodepool-name <nodepool-name> --query latestNodeImageVersion`

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

AKS cluster upgrade fails because of NSG rules

Article • 10/10/2024

This article discusses how to resolve issues if your Azure Kubernetes Service (AKS) cluster upgrade fails because of network security group (NSG) rules.

Prerequisites

This article requires Azure CLI version 2.0.65 or a later version. To find the version number, run `az --version`. If you have to install or upgrade Azure CLI, see [How to install the Azure CLI](#).

For more detailed information about the upgrade process, see the "Upgrade an AKS cluster" section in [Upgrade an Azure Kubernetes Service \(AKS\) cluster](#).

Symptoms

An AKS cluster upgrade fails, and you receive an error message that indicates that an NSG rule is involved.

Cause

An NSG rule is blocking the cluster from downloading required resources.

Solution

To resolve this issue, follow these steps:

1. Run `az network nsg list -o table`, and then locate the NSG that's linked to your cluster. The NSG is typically located in the infrastructure or node resource group, by convention named `MC_<RG name>_<your AKS cluster name>_<location code>`.
2. Run the following command to view the NSG rules:

Azure CLI

```
az network nsg rule list --resource-group <Rg name> --nsg-name <nsg name> --include-default -o table
```

The following screenshot shows the default rules.

Priority ↑↓	Name ↑↓	Port ↑↓	Protocol ↑↓	Source ↑↓	Destination ↑↓	Action ↑↓
▽ Inbound Security Rules						
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerIn...	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny
▽ Outbound Security Rules						
65000	AllowVnetOutBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowInternetOutBound	Any	Any	Any	Internet	Allow
65500	DenyAllOutBound	Any	Any	Any	Any	Deny

3. If you have the default rules, skip this step. Otherwise, revise and remove the rules that are blocking the internet traffic. Then, run the following command to upgrade the AKS cluster to the same version that you previously tried to upgrade to. This process will trigger a reconciliation.

Azure CLI

```
az aks upgrade --resource-group <ResourceGroupName> --name  
<AKSClusterName> --kubernetes-version <KUBERNETES_VERSION>
```

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

Yes

No

Memory saturation occurs in pods after cluster upgrade to Kubernetes 1.25

Article • 05/06/2025

This article discusses how to fix pods that stop working because of memory saturation or out-of-memory (OOM) errors that occur after you upgrade a Microsoft Azure Kubernetes Service (AKS) cluster to Kubernetes 1.25.x.

Symptoms

One or more of the following issues occur:

- Memory pressure on nodes
- Increased memory usage for apps when compared to their memory usage before the upgrade
- CPU throttling on nodes
- Pod failure because of OOM errors

Performance degradation can occur in apps that run in the following environments:

- Java Runtime Environment (JRE) (for [JRE versions that are earlier than version 11.0.18 or version 1.8.0 372](#))
- .NET versions that are earlier than version 5.0
- [Node.js](#)

Note

This list of environments in which performance degradation can occur isn't a comprehensive list. There might be other environments that experience memory saturation or OOM issues.

Solution

Note

If you only experience increased memory usage and no other symptoms that are mentioned in the [Symptoms](#) section, no action is needed.

Beginning in the release of Kubernetes 1.25, the [cgroup version 2 API](#) has reached general availability (GA). AKS now uses Ubuntu Linux version 22.04. By default, version 22.04 uses cgroup version 2 API. To make sure the cgroup version 2 API is available for use in other environments to prevent the memory saturation issue, follow this guidance:

- If you run Java applications, [upgrade to a Java version that supports cgroup version 2](#) and follow the guidance in [Containerize your Java applications](#). You might be able to update the base image in certain versions in which the fix has been backported. Use a version or framework that natively supports cgroup version 2. For Azure customers, Microsoft officially supports [Eclipse Temurin](#) binaries (Java 8) and [Microsoft Build of OpenJDK](#) binaries (Java 11+).
- Similarly, if you're using .NET, upgrade to [.NET version 5.0](#) or a later version.
- If you see a higher eviction rate on the pods, [use higher limits and requests for the pods](#).
- `cgroup v2` uses a different API than `cgroup v1`. If there are any applications that directly access the `cgroup` file system, update them to later versions that support `cgroup v2`. For example:
 - **Third-party monitoring and security agents:**
Some monitoring and security agents depend on the `cgroup` file system. Update these agents to versions that support `cgroup v2`.
 - **Java applications:**
Use versions that fully support `cgroup v2`:
 - OpenJDK/HotSpot: `jdk8u372`, `11.0.16`, `15`, and later versions.
 - IBM Semeru Runtimes: `8.0.382.0`, `11.0.20.0`, `17.0.8.0`, and later versions.
 - IBM Java: `8.0.8.6` and later versions.
 - **uber-go/automaxprocs:**
If you're using the `uber-go/automaxprocs` package, ensure the version is `v1.5.1` or later.
- An alternative temporary solution is to revert the `cgroup` version on your nodes by using the DaemonSet. For more information, see [Revert to cgroup v1 DaemonSet](#).

ⓘ Important

- Use the DaemonSet cautiously. Test it in a lower environment before applying to production to ensure compatibility and avoid disruptions.

- By default, the DaemonSet applies to all nodes in the cluster and reboots them to implement the `cgroup` change.
- To control how the DaemonSet is applied, configure a `nodeSelector` to target specific nodes.

Status

Microsoft is working with the Kubernetes community to resolve the issue. Track progress at [Azure/AKS Issue #3443 ↗](#).

As part of the resolution, the plan is to adjust the eviction thresholds or update [resource reservations](#), depending on the outcome of the fix.

Reference

- [Node memory usage on cgroupv2 reported higher than cgroupv1 ↗](#) (GitHub issue)

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Contact us for help

If you have questions or need help, [create a support request↗](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community ↗](#).

A load balancer with a private link service or a private link service with private endpoint connections can't be deleted

Article • 12/04/2024

This article discusses how to identify and resolve the `CannotDeleteLoadBalancerWithPrivateLinkService` or `PrivateLinkServiceWithPrivateEndpointConnectionsCannotBeDeleted` error that occurs when you try to delete a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to delete an AKS cluster, you receive one of the following error messages:

- For the `CannotDeleteLoadBalancerWithPrivateLinkService` error code:

```
internalErrorCode: "CannotDeleteLoadBalancerWithPrivateLinkService"  
StatusCode=BadRequest  
{  
    "Cannot delete load balancer  
    ..../providers/Microsoft.Network/loadBalancers/kubernetes-internal since it is  
    referenced by private link service  
    ..../providers/Microsoft.Network/privateLinkServices/test. Please delete private  
    link service prior to deleting load balancer."  
}
```

- For the `PrivateLinkServiceWithPrivateEndpointConnectionsCannotBeDeleted` error code:

```
internalErrorCode:  
"PrivateLinkServiceWithPrivateEndpointConnectionsCannotBeDeleted"  
StatusCode=BadRequest  
{
```

```
    message: "Private link service ..../Microsoft.Network/privateLinkServices/test  
cannot be deleted since it has 1 private endpoint connection. Please delete all  
private endpoint connections before deleting the private link service."  
}
```

Cause

The private link service can't be deleted because the cluster is associated with private endpoint connections.

Solution

Make sure that the private link service isn't associated with any private endpoint connections. Delete all private endpoint connections before you delete the private link service. For more information, see [Azure: How to delete a private link service that has a private endpoint connected to it?](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

A public IP address/subnet/network security group in use can't be deleted

Article • 12/04/2024

This article discusses how to identify and resolve the `PublicIPAddressCannotBeDeleted`, `InUseSubnetCannotBeDeleted`, or `InUseNetworkSecurityGroupCannotBeDeleted` error that occurs when you try to delete a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to delete an AKS cluster, you receive one of the following error messages:

- For the `PublicIPAddressCannotBeDeleted` error code:

```
{  
  
message: "Public IP address  
...../providers/Microsoft.Network/publicIPAddresses/ can not be deleted since it  
is still allocated to resource  
...../providers/Microsoft.Network/loadBalancers/kubernetes/frontendIPConfigur  
ations/..... . In order to delete the public IP, disassociate/detach the Public IP  
address from the resource."  
  
}
```

- For the `InUseSubnetCannotBeDeleted` error code:

```
{  
  
message: "Subnet aks-subnet is in use by  
...../Microsoft.Network/networkInterfaces/|providers|Microsoft.Compute|virtual  
MachineScaleSets|vmss|virtualMachines|1|networkInterfaces|aks-worker-  
vmss/ipConfigurations/ipconfig1 and cannot be deleted. In order to delete the  
subnet, delete all the resources within the subnet."  
  
}
```

or

```
{
```

```
    message: "Subnet aks-subnet is in use by  
    ..../resourceGroups/.../providers/Microsoft.Network/virtualNetworks/.../subnets  
    /.../serviceAssociationLinks/AppServiceLink and cannot be deleted. In order to  
    delete the subnet, delete all the resources within the subnet. See  
    aka.ms/deletesubnet."  
}
```

- For the `InUseNetworkSecurityGroupCannotBeDeleted` error code:

```
{  
  
    message: "Network security group  
    ..../Microsoft.Network/networkSecurityGroups/test cannot be deleted because  
    it is in use by the following resources:  
    ..../Microsoft.Network/virtualNetworks/test/subnets/test. In order to delete the  
    Network security group, remove the association with the resource(s)."  
}
```

Cause

The AKS cluster is associated with a subnet, network security group (NSG), or specific public IP address that's currently being used. This association prevents you from deleting the cluster.

Solution

- Remove all public IP addresses that are associated with Azure Load Balancer and the resource that's used by the subnet. For more information, see [View, modify settings for, or delete a public IP address](#).
- In the load balancer, remove the rules for **Load Balance rules**, **Health probes**, and **Backend pools**.
- For the NSG and subnet, remove all associated rules. For more information, see [Associate or dissociate a network security group to or from a subnet or network interface](#).
- If you're using an App Service plan with a subnet connected to the AKS cluster's VNET, you have to remove the associated App Service plan and its internal resources (such as Function App and SQL Azure database) and then retry deleting the AKS cluster.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

Troubleshoot the InUseRouteTableCannotBeDeleted error code

Article • 03/30/2025

This article discusses how to identify and resolve the `InUseRouteTableCannotBeDeleted` error that occurs when you try to delete a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to delete an AKS cluster, you receive the following error message:

```
Error Code: "InUseRouteTableCannotBeDeleted"  
{  
    "Route table aks-agentpool-routetable is in use and cannot be deleted.  
    ..../providers/Microsoft.Network/routeTables/aks-agentpool-test-routetable"  
}
```

Cause

You tried to delete the AKS cluster while its associated route table was still in use.

Solution

Remove the associated route table from the subnet. For instructions, see [Dissociate a route table from a subnet](#). Then, try again to delete the AKS cluster.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Provide product feedback ↗

Troubleshoot the LoadBalancerInUseByVirtualMachineScaleSet or NetworkSecurityGroupInUseByVirtualMachineScaleSet error code

Article • 11/21/2024

This article discusses how to identify and resolve the `LoadBalancerInUseByVirtualMachineScaleSet` or `NetworkSecurityGroupInUseByVirtualMachineScaleSet` error that occurs when you try to delete a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to delete an AKS cluster, you receive the following error message for `LoadBalancerInUseByVirtualMachineScaleSet`:

```
internalErrorCode: "LoadBalancerInUseByVirtualMachineScaleSet"  
StatusCode=409  
{  
    "Cannot delete load balancer .../Microsoft.Network/loadBalancers/kubernetes since  
    its child resources aksOutboundBackendPool, kubernetes are in use by virtual  
    machine scale set .../Microsoft.Compute/virtualMachineScaleSets/aks-worker-test-  
    vmss"  
}
```

Or, you receive the following error message for `NetworkSecurityGroupInUseByVirtualMachineScaleSet`:

```
internalErrorCode: "NetworkSecurityGroupInUseByVirtualMachineScaleSet"  
StatusCode=409  
{
```

```
"Cannot delete network security group  
.../Microsoft.Network/networkSecurityGroups/aks-agentpool since it is in use by  
virtual machine scale set .../Microsoft.Compute/virtualMachineScaleSets/aks-vmss"  
}
```

Cause

You tried to delete an AKS cluster while the virtual machine scale set was still using the associated public IP address or network security group (NSG).

Solution

To fix this issue, use one of the following methods:

- Remove all public IP addresses that are associated with Azure Load Balancer. For more information, see [View, modify settings for, or delete a public IP address](#).
- Dissociate the NSG that's used by the subnet. For more information, see [Associate or dissociate a network security group](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

Troubleshoot the LoadBalancerInUseByVirtualMachineScaleSet or NetworkSecurityGroupInUseByVirtualMachineScaleSet error code

Article • 11/21/2024

This article discusses how to identify and resolve the `LoadBalancerInUseByVirtualMachineScaleSet` or `NetworkSecurityGroupInUseByVirtualMachineScaleSet` error that occurs when you try to delete a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to delete an AKS cluster, you receive the following error message for `LoadBalancerInUseByVirtualMachineScaleSet`:

```
internalErrorCode: "LoadBalancerInUseByVirtualMachineScaleSet"  
StatusCode=409  
{  
    "Cannot delete load balancer .../Microsoft.Network/loadBalancers/kubernetes since  
    its child resources aksOutboundBackendPool, kubernetes are in use by virtual  
    machine scale set .../Microsoft.Compute/virtualMachineScaleSets/aks-worker-test-  
    vmss"  
}
```

Or, you receive the following error message for `NetworkSecurityGroupInUseByVirtualMachineScaleSet`:

```
internalErrorCode: "NetworkSecurityGroupInUseByVirtualMachineScaleSet"  
StatusCode=409  
{
```

```
"Cannot delete network security group  
.../Microsoft.Network/networkSecurityGroups/aks-agentpool since it is in use by  
virtual machine scale set .../Microsoft.Compute/virtualMachineScaleSets/aks-vmss"  
}
```

Cause

You tried to delete an AKS cluster while the virtual machine scale set was still using the associated public IP address or network security group (NSG).

Solution

To fix this issue, use one of the following methods:

- Remove all public IP addresses that are associated with Azure Load Balancer. For more information, see [View, modify settings for, or delete a public IP address](#).
- Dissociate the NSG that's used by the subnet. For more information, see [Associate or dissociate a network security group](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

Troubleshoot the TooManyRequestsReceived or SubscriptionRequestsThrottled error code

Article • 04/03/2025

This article discusses how to identify and resolve the `TooManyRequestsReceived` or `SubscriptionRequestsThrottled` error that occurs when you try to delete a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to delete an AKS cluster, you receive the following error message:

```
internalErrorCode: TooManyRequestsReceived
```

```
StatusCode: 429
```

```
{
```

```
message: "Number of read requests for subscription '....'" exceeded the limit of '....' for time interval 'XX:XX:XX'. Please try again after '....' seconds."
```

```
}
```

Cause

Every subscription-level and tenant-level operation is subject to throttling limits. These limits apply to each instance of Azure Resource Manager. When you reach the limit, you receive an HTTP response that indicates status code 429: "Too many requests."

Solution

The HTTP response includes a `Retry-After` value. This specifies the number of seconds that your application should wait (or sleep) before it sends the next request. If you send a request before the retry value has elapsed, your request isn't processed, and a new retry value is returned. For more information about throttling limits, see [Throttling Resource Manager requests](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the TooManyRequestsReceived or SubscriptionRequestsThrottled error code

Article • 04/03/2025

This article discusses how to identify and resolve the `TooManyRequestsReceived` or `SubscriptionRequestsThrottled` error that occurs when you try to delete a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to delete an AKS cluster, you receive the following error message:

```
internalErrorCode: TooManyRequestsReceived
```

```
StatusCode: 429
```

```
{
```

```
message: "Number of read requests for subscription '....'" exceeded the limit of '....' for time interval 'XX:XX:XX'. Please try again after '....' seconds."
```

```
}
```

Cause

Every subscription-level and tenant-level operation is subject to throttling limits. These limits apply to each instance of Azure Resource Manager. When you reach the limit, you receive an HTTP response that indicates status code 429: "Too many requests."

Solution

The HTTP response includes a `Retry-After` value. This specifies the number of seconds that your application should wait (or sleep) before it sends the next request. If you send a request before the retry value has elapsed, your request isn't processed, and a new retry value is returned. For more information about throttling limits, see [Throttling Resource Manager requests](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Cluster autoscaler fails to scale with "cannot scale cluster autoscaler enabled node pool" error

07/16/2025

This article discusses how to resolve the "cannot scale cluster autoscaler enabled node pool" error that appears when scaling a cluster with an autoscaler enabled node pool.

Symptoms

You receive an error message that resembles the following message:

`kubectl get nodes` outputs "No resources found"

All pods state is `Pending`

Scale operations are failing with "Cannot scale cluster autoscaler enabled node pool" error

Troubleshooting checklist

Azure Kubernetes Service (AKS) uses virtual machine scale sets-based agent pools, which contain cluster nodes and [cluster autoscaling capabilities](#) if enabled.

Check that the cluster virtual machine scale set exists

1. Sign in to [Azure portal](#).

2. Find the node resource group by searching the following names:

- The default name

`MC_{AksResourceGroupName}_{YourAksClusterName}_{AksResourceLocation}`.

- The custom name (if it was provided at creation).

Note

When you create a new cluster, AKS automatically creates a second resource group to store the AKS resources. For more information, see [Why are two resource groups created with AKS?](#)

3. Check the list of resources and make sure that there's a virtual machine scale set.

Cause 1: The cluster virtual machine scale set was deleted

Deleting the virtual machine scale set attached to the cluster causes the cluster autoscaler to fail. It also causes issues when provisioning resources such as nodes and pods.

! Note

Modifying any resource under the node resource group in the AKS cluster is an unsupported action and will cause cluster operation failures. You can prevent changes from being made to the node resource group by [blocking users from modifying resources](#) managed by the AKS cluster.

Reconcile node pool

If the cluster virtual machine scale set is accidentally deleted, you can reconcile the node pool by using `az aks nodepool update`:

shell

```
# Update Node Pool Configuration
az aks nodepool update --resource-group <resource-group-name> --cluster-name
<cluster-name> --name <nodepool-name> --tags <tags> --node-taints <taints> --
labels <labels>

# Verify the Update
az aks nodepool show --resource-group <resource-group-name> --cluster-name
<cluster-name> --name <nodepool-name>
```

Monitor the node pool to make sure that it's functioning as expected and that all nodes are operational.

Cause 2: Tags or any other properties were modified from the node resource group

You may receive scaling errors if you modify or delete Azure-created tags and other resource properties in the node resource group. For more information, see [Can I modify tags and other properties of the AKS resources in the node resource group?](#)

Reconcile node resource group tags

Use the Azure CLI to make sure that the node resource group has the correct tags for AKS name and the AKS group name:

shell

```
# Add or update tags for AKS name and AKS group name
az group update --name <node-resource-group-name> --set tags.AKS-Managed-Cluster-
Name=<aks-managed-cluster-name> tags.AKS-Managed-Cluster-RG=<aks-managed-cluster-
rg>

# Verify the tags
az group show --name <node-resource-group-name> --query "tags"
```

Monitor the resource group to make sure that the tags are correctly applied and that the resource group is functioning as expected.

Cause 3: The cluster node resource group was deleted

Deleting the cluster node resource group causes issues when provisioning the infrastructure resources required by the cluster, which causes the cluster autoscaler to fail.

Solution: Update the cluster to the goal state without changing the configuration

To resolve this issue, you can run the following command to recover the deleted virtual machine scale set or any tags (missing or modified):

 Note

It might take a few minutes until the operation completes.

Set your environment variables for the AKS cluster resource group and cluster name before running the command. A random suffix is included to prevent name collisions during repeatable executions, but you must ensure the resource group and cluster exist.

Azure CLI

```
export RANDOM_SUFFIX=$(head -c 3 /dev/urandom | xxd -p)
export AKS_RG_NAME="MyAksResourceGroup$RANDOM_SUFFIX"
```

```
export AKS_CLUSTER_NAME="MyAksCluster$RANDOM_SUFFIX"  
az aks update --resource-group $AKS_RG_NAME --name $AKS_CLUSTER_NAME --no-wait
```

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Common issues when you run or scale large AKS clusters FAQ

Article • 09/27/2024

This article answers frequently asked questions about common issues that might occur when you run or scale large clusters in Microsoft Azure Kubernetes Service (AKS). A large cluster is any cluster that runs at more than a 500-node scale.

I get a "quota exceeded" error during creation, scale up, or upgrade

To resolve this issue, create a support request in the subscription in which you're trying to create, scale, or upgrade, and request a quota for the corresponding resource type. For more information, see [regional compute quotas](#).

I get an "insufficientSubnetSize" error when I deploy an AKS cluster that uses advanced networking

This error indicates that a subnet that's in use for a cluster no longer has available IPs within its CIDR for successful resource assignment. This issue can occur during upgrades, scale outs, or node pool creation. This issue occurs because the number of free IPs in the subnet is less than the result of the following formula:

number of nodes requested * node pool `--max-pod` value

Prerequisites

- To scale beyond 400 nodes, you have to use the [Azure CNI networking plug-in](#).
- To help plan your virtual network and subnets to accommodate the number of nodes and pods that you're deploying, see [planning IP addresses for your cluster](#). To reduce the overhead of subnet planning or cluster re-creation that you would do because of IP exhaustion, see [Dynamic IP allocation](#).

Solution

Because you can't update an existing subnet's CIDR range, you must have permission to create a new subnet to resolve this issue. Follow these steps:

1. Rebuild a new subnet that has a larger CIDR range that's sufficient for operation goals.
2. Create a new subnet that has a new non-overlapping range.
3. Create a new node pool on the new subnet.
4. Drain pods from the old node pool that resides in the old subnet that will be replaced.
5. Delete the old subnet and old node pool.

I'm having sporadic egress connectivity failures because of SNAT port exhaustion

For clusters that run at a relatively large scale (more than 500 nodes), we recommend that you use the AKS [Managed Network Address Translation \(NAT\) Gateway](#) for greater scalability. Azure NAT Gateway allows up to 64,512 outbound UDP and TCP traffic flows per IP address, and a maximum of 16 IP addresses.

If you're not using Managed NAT, see [Troubleshoot source network address translation \(SNAT\) exhaustion and connection timeouts](#) to understand and resolve SNAT port exhaustion issues.

I can't scale up to 5,000 nodes using the Azure portal

Use the Azure CLI to scale up to a maximum of 5,000 nodes by following these steps:

1. Create a minimum number of node pools in the cluster (because the maximum node pool node limit is 1,000) by running the following command:

Bash

```
az aks nodepool add --resource-group MyResourceGroup --name nodepool1 -  
-cluster-name MyManagedCluster
```

2. Scale up the node pools one at a time. Ideally, set five minutes of sleep time between consecutive scale-ups of 1,000. Run the following command:

Bash

```
az aks nodepool scale --resource-group MyResourceGroup --name nodepool1  
--cluster-name MyManagedCluster
```

My upgrade is running, but it's slow

In its default configuration, AKS surges during an upgrade by taking the following actions:

- Creating one new node.
- Scaling the node pool beyond the desired number of nodes by one node.

For the max surge settings, a default value of one node means that AKS creates one new node before it drains the existing applications and replaces an earlier-versioned node. This extra node lets AKS minimize workload disruption.

When you upgrade clusters that have many nodes, it can take several hours to upgrade the entire cluster if you use the default value of `max-surge`. You can customize the `max-surge` property per node pool to enable a tradeoff between upgrade speed and upgrade disruption. By increasing the max surge value, you enable the upgrade process to finish sooner. However, a large value for max surge might also cause disruptions during the upgrade process.

Run the following command to increase or customize the max surge for an existing node pool:

Bash

```
az aks nodepool update --resource-group MyResourceGroup --name mynodepool --  
cluster-name MyManagedCluster --max-surge 5
```

It's also important to consider how your deployment settings might delay the completion of the upgrade or scale operation:

- [SKU family B series VMs are not supported](#) by AKS in the system nodepool and they can experience low performance during and after updates.
- Check your deployment's PDB resource settings to ensure they are accurate for a successful upgrade. For more information, see [AKS workload best practices](#).

 Tip

To get more insights about this behavior, you can [view error details on the Activity Log page in the Azure portal](#) or review the [resource logs](#) on your cluster.

My upgrade is reaching the quota (5,000 cluster) limit

To resolve this issue, see [Increase regional vCPU quotas](#).

My internal service creation at more than 750 nodes is slow or failing because of a time-out error

Standard Load Balancer (SLB) back-end pool updates are a known performance bottleneck. We're working on a new capability that will enable faster creation of services and SLB at scale. To send us your feedback about this issue, see [Azure Kubernetes support for load balancer with IP-based back-end pool](#).

Solution

We recommend that you scale down the cluster to fewer than 750 nodes, and then create an internal load balancer for the cluster. To create an internal load balancer, create a `LoadBalancer` service type and `azure-load-balancer-internal` annotation, per the following example procedure.

Step 1: Create an internal load balancer

To create an internal load balancer, create a service manifest that's named `internal-lb.yaml` and that contains the `LoadBalancer` service type and the `azure-load-balancer-internal` annotation, as shown in the following example:

YAML

```
apiVersion: v1
kind: Service
metadata:
  name: internal-app
  annotations:
    service.beta.kubernetes.io/azure-load-balancer-internal: "true"
spec:
  type: LoadBalancer
```

```
ports:  
- port: 80  
selector:  
  app: internal-app
```

Step 2: Deploy the internal load balancer

Deploy the internal load balancer by using the `kubectl apply` command, and specify the name of your YAML manifest, as shown in the following example:

Bash

```
kubectl apply -f internal-lb.yaml
```

After your cluster is created, you can also provision an internal load balancer (per this procedure), and keep an internal load-balanced service running. Doing this enables you to add more services to the load balancer at scale.

SLB service creation at scale takes hours to run

SLB back-end pool updates are a known performance bottleneck. We're working on a new capability that will allow you to run load balanced services at scale with considerably faster performance for create, update, and delete operations. To send us your feedback, see [Azure Kubernetes support for load balancer with IP-based back-end pool](#).

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Cluster autoscaler fails to scale with "failed to fix node group sizes" error

Article • 10/09/2024

This article discusses how to resolve the "failed to fix node group sizes" error that appears on the cluster autoscaler logs when you're creating or managing AKS clusters.

Symptoms

Your cluster autoscaler isn't scaling up or down, and you see an error similar to the following error on the [cluster autoscaler logs](#).

Output

```
E1114 09:58:55.367731 1 static_autoscaler.go:239] Failed to fix node group sizes: failed to decrease aks-default-35246781-vmss: attempt to delete existing nodes
```

Cause

This error is caused by an upstream cluster autoscaler race condition. In such a case, cluster autoscaler gets stuck in a deadlock. For more information, see [cluster-autoscaler gets stuck with "Failed to fix node group sizes" error](#).

Solution

To get out of this state, disable and re-enable the [cluster autoscaler](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?



Yes



No

Troubleshoot the SubnetIsFull error code

Article • 03/26/2025

This article discusses how to identify and resolve the `SubnetIsFull` error that occurs when you try to scale a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- [Azure CLI](#) (version 2.0.59 or a later version)

Symptoms

When you try to scale an AKS cluster, you receive the following error message:

```
"code": "SubnetIsFull"  
  
"message": "Subnet <subnet-name> with address prefix <subnet-prefix> does not  
have enough capacity for <new-ip-count> IP addresses."
```

Cause

To add nodes to an AKS cluster (scaling out), you have to use more IP addresses from the subnet in which the node pool is deployed. The exact number of new IP addresses that are required to successfully complete a cluster scale operation varies according to the networking plug-in that the cluster uses. For information about how IP addresses are allocated under each of these networking models, see [Network concepts for applications in AKS](#).

Note

Azure reserves five IP addresses per subnet. The first address in the subnet is for the network ID, followed by three addresses that are used internally by Azure. The last address in the subnet is reserved for broadcast packets. For more information, see [Are there any restrictions on using IP addresses within these subnets?](#)

Solution

Trying to update a subnet's Classless Inter-Domain Routing (CIDR) address space in an existing node pool isn't currently supported. To migrate your workloads to a new node pool in a larger subnet, follow these steps:

1. Create a subnet in the cluster virtual network that contains a larger CIDR address range than the existing subnet. For information about how to adequately size the subnet for your cluster, see [Plan IP addressing for your cluster](#).
2. Create a node pool on the new subnet by running the `az aks nodepool add` command together with the `--vnet-subnet-id` parameter.
3. Migrate your workloads to the new node pool by draining the nodes in the old node pool. For information about how to safely drain AKS worker nodes, see [Safely Drain a Node](#).
4. Delete the original node pool by running the `az aks nodepool delete` command.

Best practices

To avoid `SubnetIsFull` issues in Azure Kubernetes Service (AKS), follow best practices that are related to subnet sizing, IP address management, and node pool strategies. Here are some key recommendations:

- Plan for Future Growth: When you create subnets, make sure that they're large enough to accommodate future growth. We recommend that you reserve more IP addresses than you currently need to avoid running out of space as the cluster scales.
- Use Larger Subnet CIDR: If possible, use a larger subnet CIDR to provide more IP addresses. This strategy helps to accommodate more nodes and pods without running into IP exhaustion issues.
- Monitor IP Usage: To identify potential issues before they become critical, regularly monitor the IP address usage within your subnets. Tools such as Azure Monitor can help track IP address consumption.
- Optimize IP Allocation: Make sure that IP addresses are allocated efficiently. Avoid reserving IP addresses unnecessarily. To free up space, release any unused IP addresses.
- Use multiple node pools: Consider using node pools that have different subnets to distribute the IP address load. This strategy can help mitigate the risk of running out of IP addresses in a single subnet.

More information

- General troubleshooting of AKS cluster creation issues

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?



[Provide product feedback](#)

Troubleshoot cluster upgrading and scaling errors

Article • 09/19/2024

This article discusses how to troubleshoot errors that occur when you try to upgrade or scale a Microsoft Azure Kubernetes Service (AKS) cluster.

Some causes of failure when you try to upgrade or scale an AKS cluster are as follows.

Cause 1: Cluster is in a failed state

If a cluster is in a `failed` state, `upgrade` or `scale` operations won't succeed. A cluster can enter a failed state for many reasons.

Here are the most common reasons and corresponding solutions:

- Scaling while having an insufficient Compute Resource Provider (CRP) quota.

To resolve this issue, increase your resource quota before you scale by following these steps:

1. Scale your cluster back to a stable goal state within the quota.
2. [Request an increase in your resource quota](#).
3. Try to scale up again beyond the initial quota limits.
4. Retry the original operation. This second operation should bring your cluster to a successful state.

- Scaling a cluster that uses advanced networking such as Azure Container Networking Interface (CNI), Azure CNI for dynamic IP allocation but has insufficient subnet (networking) resources.

To resolve this issue, see [Troubleshoot the SubnetIsFull error code](#).

- Upgrading a cluster that has Pod Disruption Budgets (PDBs) which may cause eviction failures.

To resolve this issue, remove or adjust the PDB so that the pod can be drained. For more information, see [Troubleshoot UpgradeFailed errors due to eviction failures caused by PDBs](#).

- Upgrading a cluster that uses deprecated APIs.

For Kubernetes versions upgrading to 1.26 or later, AKS checks whether deprecated APIs are used before starting the cluster upgrade. To resolve this issue and start to upgrade, see [How to mitigate stopped upgrade operations due to deprecated APIs](#).

Cause 2: You're trying to upgrade and scale at the same time

A cluster or node pool can't simultaneously upgrade and scale. Instead, each operation type must finish on the target resource before the next request runs on that same resource. Therefore, operations are limited when active upgrade or scale operations are occurring or attempted.

To resolve this issue, follow these steps:

1. Determine the current status of your cluster before you try an operation.

To retrieve detailed status about your cluster, run the following `az aks show` command:

Azure CLI

```
az aks show --resource-group myResourceGroup --name myAKSCluster --output table
```

2. Refer to the following table to take the appropriate action based on the cluster's status:

[+] Expand table

ProvisioningState	Action
Upgrading	Wait until the operation finishes.
Failed	Follow the solutions that are outlined in Cause 1 .
Succeeded	Retry the scale or other previously failed operation.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Current node count is outside the autoscaler min and max range

Article • 03/25/2025

Symptoms

You're using the autoscaler for a Microsoft Azure Kubernetes Service (AKS) cluster. When you restart your cluster, your current node count doesn't fall within the minimum and maximum range of values that you set.

Cause

This behavior is expected. The cluster starts by setting the number of nodes that it needs to run its workloads. These values aren't affected by your autoscaler settings. When your cluster does scaling operations, the minimum and maximum values will affect your current node count.

For more information about this behavior, see [My cluster is below minimum / above maximum number of nodes, but CA did not fix that! Why?](#)

More information

Your cluster will eventually enter and remain in the desired range until you stop the cluster.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?



Yes



No

[Provide product feedback](#)

Basic troubleshooting of AKS cluster startup issues

Article • 03/14/2025

This article outlines the basic troubleshooting methods to use if you can't start a Microsoft Azure Kubernetes Service (AKS) cluster successfully.

Prerequisites

- [Azure CLI](#) (version 2.0.59 or a later version).
- The Kubernetes [kubectl](#) tool. To install kubectl by using Azure CLI, run the [az aks install-cli](#) command.

View errors from Azure CLI

When you start clusters by using Azure CLI, errors are recorded as output if the operation fails. Here's how a command, user input, and operation output might appear in a `Bash` console:

```
Console

$ az aks start --resource-group myResourceGroup --name MyManagedCluster
(VMExtensionProvisioningError) Unable to establish outbound connection from
agents, please see https://learn.microsoft.com/en-
us/troubleshoot/azure/azure-kubernetes/error-code-
outboundconnfailvmextensionerror and https://aka.ms/aks-required-ports-and-
addresses for more information.
Details: instance 3 has extension error details : {vmssCSE error messages :
{vmssCSE exit status=50, output=AGE_SHA=sha-16fd35}
```

These errors often contain detailed descriptions of what went wrong in the cluster start operation, and they provide links to articles that contain more details. Additionally, you can use our troubleshooting articles as a reference based on the error that an Azure CLI operation produces.

View error details in the Azure portal

To view the details about errors in the [Azure portal](#), examine the [Azure activity log](#). To find the list of activity logs in the Azure portal, search on **Activity log**. Or, select

Notifications (the bell icon), and then select **More events in the activity log**.

The list of logs on the **Activity log** page contains a line entry in which the **Operation name** column value is named **Start Managed Cluster**. The corresponding **Event initiated by** column value is set to the name of your work or school account. If the operation is successful, the **Status** column value shows **Accepted**.

A screenshot of the Azure Activity Log page. The top navigation bar includes 'Activity', 'Edit columns', 'Refresh', 'Export Activity Logs', 'Download as CSV', and 'Insights'. A banner at the top says 'Looking for Log Analytics? In Log Analytics you can search for performance, diagnostics, health logs, and more...'. Below the banner is a search bar and a 'Quick Insights' button. Filter options include 'Management Group : None', 'Subscription : j...', 'Event severity : All', and 'Timespan :'. A 'Add Filter' button is also present. The main area shows a table with 5 items:

Operation name	Status	Time	Time stamp
> ! 'deployIfNotExists' Policy action.	Failed	10 minutes ...	Mon Jan 22 ...
▽ ! Start Managed Cluster	Failed	23 minutes ...	Mon Jan 22 ...
! Start Managed Cluster	Started	36 minutes ...	Mon Jan 22 ...
! Start Managed Cluster	Accepted	36 minutes ...	Mon Jan 22 ...
> ! Stop Managed Cluster	Succeeded	37 minutes ...	Mon Jan 22 ...

What if an error occurred instead? In that case, the **Start Managed Cluster** operation **Status** field shows **Failed**. Unlike in the operations to create cluster components, here you must expand the failed operation entry to review the suboperation entries. Typical suboperation names are policy actions, such as **'audit' Policy action** and **'auditIfNotExists' Policy action**. Some suboperations will continue to show that they succeeded.

To further investigate, you can select one of the failed suboperations. A side pane opens so that you can review more information about the suboperation. You can troubleshoot values for fields such as **Summary**, **JSON**, and **Change history**. The **JSON** field contains the output text for the error in JSON format, and it usually provides the most helpful information.

Start Managed Cluster

Mon Jan 22 2024 14:45:49 GMT-0800 (Pacific Standard Time)

[+ New alert rule](#) [+ New support request](#)

[Summary](#) [JSON](#) [Change history](#)

```
82 |     "statusMessage": "{\"status\":\"Failed\",\"error\":\":\n|       \\"code\\\":\"ResourceOperationFailure\\\",\\\"message\\\":\"The resource operation completed with\n|       terminal provisioning state 'Failed'.\\\",\\\"details\\\":[\n|           \\"code\\\":\"VMExtensionProvisioningError\\\",\\\"message\\\":\"Unable to establish outbound\n|           connection from agents, please see https://learn.microsoft.com/en-us/troubleshoot/azure/azure-kubernetes/error-code-outboundconnfailvmextensionerror and https://aka.ms/aks-required-ports-and-addresses for more information. Details: instance 2 has extension\n|           error details : {vmssCSE error messages : {vmssCSE exit status=50,\n|               output=AGE_SHA=sha-16fd35\\n++ NVIDIA_DRIVER_IMAGE_SHA=sha-16fd35\\n++ export\n|               NVIDIA_DRIVER_IMAGE_TAG=cuda-525.85.12-sha-16fd35\\n++ NVIDIA_DRIVER_IMAGE_TAG=cuda-525.85.\n|               12-sha-16fd35\\n++ export NVIDIA_DRIVER_IMAGE=mcr.microsoft.com/aks/aks-gpu\\n++\n|               NVIDIA_DRIVER_IMAGE=mcr.microsoft.com/aks/aks-gpu\\n++ export 'CTR_GPU_INSTALL_CMD=ctr run\n|               --privileged --rm --net-host --with-ns pid:/proc/1/ns/pid --mount type=bind,src=/opt/gpu,\n|               dst=/mnt/gpu,options=rbind --mount type=bind,src=/opt/actions,dst=/mnt/actions,\n|               options=rbind'\\n++ CTR_GPU_INSTALL_CMD='ctr run --privileged --rm --net-host --with-ns\n|               pid:/proc/1/ns/pid --mount type=bind,src=/opt/gpu,dst=/mnt/gpu,options=rbind --mount\n|               type=bind,src=/opt/actions,dst=/mnt/actions,options=rbind'\\n++ export\n|               'DOCKER_GPU_INSTALL_CMD=docker run --privileged --net=host --pid=host -v /opt/gpu:/mnt/gpu\n|               -v /opt/actions:/mnt/actions --rm'\\n++ DOCKER GPU INSTALL CMD='docker run --privileged\n|           
```

View cluster insights

You can also generate cluster insights to help you troubleshoot via the **Diagnose and solve problems** blade in the Azure portal. To access this feature, follow these steps:

1. In [the Azure portal](#), search for and select **Kubernetes services**.
2. Select the name of your AKS cluster.
3. In the navigation pane of the AKS cluster page, select **Diagnose and solve problems**.
4. On the **Diagnose and solve problems** page, select the **Cluster insights** link. The cluster insights tool analyzes your cluster, and then provides a list of its findings in the **Observations and Solutions** section of the **Cluster Insights** page.
5. Select one of the findings to view more information about a problem and its possible solutions.

View resources in the Azure portal

In the Azure portal, you might want to view the resources that were created when the cluster was built. Typically, these resources are in a resource group that begins with *MC_*. The managed cluster resource group might have a name such as

`MC_MyResourceGroup_MyManagedCluster_<location-code>`. However, the name may be different if you built the cluster by using a custom-managed cluster resource group.

To find the resource group, search for and select **Resource groups** in the Azure portal, and then select the resource group in which the cluster was created. The resource list is shown on the **Overview** page of the resource group.

 **Warning**

We recommend that you don't modify resources in the `MC_` resource group. This action might cause unwanted effects on your AKS cluster.

To review the status of a virtual machine scale set, you can select the scale set name within the resource list of the resource group. It might have a **Name** similar to `aks-nodepool1-12345678-vmss` and a **Type** value of **Virtual machine scale set**. The status of the scale set appears at the top of the node pool's **Overview** page, and more details are shown in the **Essentials** heading. If the deployment was unsuccessful, the displayed status is **Failed**.

For all resources, you can review details to better understand why the deployment failed. For a scale set, you can select the **Failed** status text to view details about the failure. The details are in a row that contains the **Status**, **Level**, and **Code** columns. The following example shows a row of column values.

 Expand table

Column	Example value
Status	Provisioning failed
Level	Error
Code	ProvisioningState/failed/VMExtensionProvisioningError

Select the row to see the **Message** field. This contains even more information about that failure. For example, the **Message** field for the example row begins with the following text:

VM has reported a failure when processing extension 'vmssCSE'. Error message:
"Enable failed: failed to execute command: command terminated with exit status=50
[stdout] [stderr] 0 0 0 --: Armed with this information, you can conclude that the VMs in the scale set failed and generated exit status 50.

Use kubectl commands

For another option to help troubleshoot errors on your cluster, enter kubectl commands to get details about the resources that were deployed in the cluster. To use kubectl, first sign in to your AKS cluster:

Azure CLI

```
az aks get-credentials --resource-group MyResourceGroup --name  
MyManagedCluster
```

Depending on the type of failure and when it occurred, you might not be able to sign in to your cluster to get more details. But in general, if your cluster was created and shows up in the Azure portal, you should be able to sign in and run kubectl commands.

View cluster nodes (kubectl get nodes)

To get more details to determine the state of the nodes, view the cluster nodes by entering the [kubectl get](#) nodes command. In this example, no nodes are reporting in the cluster:

Console

```
$ kubectl get nodes  
No resources found
```

View pods in the system namespace (kubectl get pods)

Viewing the pods in the kube-system namespace is also a good way to troubleshoot your issue. This method lets you view the status of the Kubernetes system pods. In this example, we enter the `kubectl get pods` command:

Console

```
$ kubectl get pods -n kube-system  
NAME                               READY   STATUS    RESTARTS   AGE  
coredns-845757d86-7xjqb           0/1     Pending   0          78m  
coredns-autoscaler-5f85dc856b-mxkrj 0/1     Pending   0          77m  
konnnectivity-agent-67f7f5554f-nsw2g 0/1     Pending   0          77m  
konnnectivity-agent-8686cb54fd-xlsgk 0/1     Pending   0          65m  
metrics-server-6bc97b47f7-dfhbr    0/1     Pending   0          77m
```

Describe the status of a pod (kubectl describe pod)

By describing the status of the pods, you can view the configuration details and any events that have occurred on the pods. Run the [kubectl describe](#) pod command:

```
Console

$ kubectl describe pod coredns-845757d86-7xjqb -n kube-system
Name:           coredns-845757d86-7xjqb
Namespace:      kube-system
Priority:      2000001000
Priority Class Name: system-node-critical
Node:          <none>
Labels:         k8s-app=kube-dns
                kubernetes.io/cluster-service=true
                pod-template-hash=845757d86
                version=v20
...
Events:
  Type     Reason        Age           From            Message
  ----     -----        ---          ----           -----
  Warning  FailedScheduling  24m (x1 over 25m)  default-scheduler  no nodes
available to schedule pods
  Warning  FailedScheduling  29m (x57 over 84m)  default-scheduler  no nodes
available to schedule pods
```

In the command output, you can see that the pod can't be deployed to a node because no nodes are available.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

Troubleshoot the K8SAPIServerConnFailVMExtensionError error code (51)

Article • 03/12/2025

This article discusses how to identify and resolve the `K8SAPIServerConnFailVMExtensionError` error (also known as error code `ERR_K8S_API_SERVER_CONN_FAIL`, error number 51) that occurs when you try to start or create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- The [Netcat](#) (nc) command-line tool

Symptoms

When you try to start or create an AKS cluster, you receive the following error message:

Unable to establish connection from agents to Kubernetes API server, please see <https://aka.ms/aks-required-ports-and-addresses> for more information.

Details: Code="VMExtensionProvisioningError"

Message="VM has reported a failure when processing extension 'vmssCSE'.

Error message: "Enable failed: failed to execute command: command terminated with exit status=51\n[stdout]\n{

"ExitCode": "51",

"Output": "Thu Oct 14 18:07:37 UTC 2021,aks-nodepool1-18315663-vmss000000\nConnection to

Cause

Your cluster nodes can't connect to your cluster API server pod.

Solution

Run a Netcat command to verify that your nodes can resolve the cluster's fully qualified domain name (FQDN):

```
shell
```

```
nc -vz <cluster-fqdn> 443
```

If you're using egress filtering through a firewall, make sure that traffic is allowed to your cluster FQDN.

In rare cases, the firewall's outbound IP address can be blocked if you've authorized IP addresses that are enabled on your cluster. In this scenario, you must add the outbound IP address of your firewall to the list of authorized IP ranges for the cluster. For more information, see [Secure access to the API server using authorized IP address ranges in AKS](#).

More information

- [General troubleshooting of AKS cluster creation issues](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

Troubleshoot the K8SAPIServerDNSLookupFailVMExtensionError error code (52)

07/08/2025

This article discusses how to identify and resolve the `K8SAPIServerDNSLookupFailVMExtensionError` error (also known as error code `ERR_K8S_API_SERVER_DNS_LOOKUP_FAIL`, error number 52) that occurs when you try to start or create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- The [nslookup](#) DNS lookup tool for Windows nodes or the [dig](#) tool for Linux nodes.
- [Azure CLI](#), version 2.0.59 or a later version. If Azure CLI is already installed, you can find the version number by running `az --version`.

Symptoms

When you try to start or create an AKS cluster, you receive the following error message:

Agents are unable to resolve Kubernetes API server name. It's likely custom DNS server is not correctly configured, please see <https://aka.ms/aks/private-cluster#hub-and-spoke-with-custom-dns> for more information.

Details: Code="VMExtensionProvisioningError"

Message="VM has reported a failure when processing extension 'vmssCSE'.

Error message: "Enable failed: failed to execute command: command terminated with exit status=52\n[stdout]\n{

"ExitCode": "52",

"Output": "Fri Oct 15 10:06:00 UTC 2021,aks- nodepool1-36696444-vmss000000\\nConnection to mcr.microsoft.com 443 port [tcp/https]

Cause

The cluster nodes can't resolve the cluster's fully qualified domain name (FQDN) in Azure DNS. Run the following DNS lookup command on the failed cluster node to find DNS resolutions that are valid.

[+] [Expand table](#)

Node OS	Command
Linux	<code>dig <cluster-fqdn></code>
Windows	<code>nslookup <cluster-fqdn></code>

Solution

On your DNS servers and firewall, make sure that nothing blocks the resolution to your cluster's FQDN. Your custom DNS server might be incorrectly configured if something is blocking even after you run the `nslookup` or `dig` command and apply any necessary fixes. For help to configure your custom DNS server, review the following articles:

- [Create a private AKS cluster](#)
- [Private Azure Kubernetes service with custom DNS server ↗](#)
- [What is IP address 168.63.129.16?](#)

When you use a private cluster that has a custom DNS, a DNS zone is created. The DNS zone must be linked to the virtual network. This occurs after the cluster is created. Creating a private cluster that has a custom DNS fails during creation. However, you can restore the creation process to a "success" state by reconciling the cluster. To do this, run the `az resource update` command in Azure CLI, as follows:

Below, set your AKS cluster and resource group names, then run the update command to reconcile the cluster. The environment variables will make your resource names unique and are declared just before use.

Azure CLI

```
az resource update --resource-group $RESOURCE_GROUP_NAME \
--name $CLUSTER_NAME \
--namespace Microsoft.ContainerService \
--resource-type ManagedClusters
```

Results:

Output

```
{  
  "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-  
xxxxxxxxxxxx/resourceGroups/myResourceGroupxxx/providers/Microsoft.ContainerService/ManagedClusters/myAksClusterxxx",  
  "location": "eastus",  
  "name": "myAksClusterxxx",  
  "properties": {  
    // ...other properties...  
  },  
  "resourceGroup": "myResourceGroupxxx",  
  "type": "Microsoft.ContainerService/ManagedClusters"  
}
```

Also verify that your DNS server is configured correctly for your private cluster, as described earlier.

 **Note**

Conditional Forwarding doesn't support subdomains.

More information

- [General troubleshooting of AKS cluster creation issues](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the OutboundConnFailVMExtensionError error code (50)

07/24/2025

This article describes how to identify and resolve the `OutboundConnFailVMExtensionError` error (also known as error code `ERR_OUTBOUND_CONN_FAIL`, error number 50) that might occur if you create, start, scale, or upgrade a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- The [Netcat](#) (nc) command-line tool
- The [dig](#) command-line tool
- The Client URL ([cURL](#)) tool

Symptoms

When you try to create, scale, or upgrade an AKS cluster, you may receive the following error message:

Unable to establish outbound connection from agents, please see <https://aka.ms/aks-required-ports-and-addresses> for more information.

Details: Code="VMExtensionProvisioningError"

Message="VM has reported a failure when processing extension 'vmssCSE'.

Error message: "Enable failed: failed to execute command: command terminated with exit status=50\n[stdout]\n[nstderr]\nncc: connect to mcr.microsoft.com port 443 (tcp) failed: Connection timed out\nCommand exited with non-zero status

Error details : "vmssCSE error messages : {vmssCSE exit status=50, output=pt/apt.conf.d/95proxy...}

This error may also cause your running nodes to become NotReady or cause image pull failures because outbound connectivity is blocked from some or all nodes in your cluster.

Cause

The custom script extension that downloads the necessary components to provision the nodes couldn't establish the necessary outbound connectivity to obtain packages. For public clusters, the nodes try to communicate with the Microsoft Container Registry (MCR) endpoint (`mcr.microsoft.com`) on port 443.

There are many reasons why the outbound traffic might be blocked. The best way to troubleshoot outbound connectivity failures is by running a connectivity analysis with [Azure Virtual Network Verifier \(Preview\)](#). By running a connectivity analysis, you can visualize the hops within the traffic flow and any misconfigurations within Azure networking resources that are blocking traffic. To manually troubleshoot outbound connectivity failures, you can use the Secure Shell protocol (SSH) to connect to the node. This section covers instructions for both types of investigations:

Check if Azure network resources are blocking traffic to the endpoint

To determine if traffic is blocked to the endpoint due to Azure network resources, run a connectivity analysis from your AKS cluster nodes to the endpoint using the [Azure Virtual Network Verifier \(Preview\)](#) tool. The connectivity analysis covers the following resources:

- Azure Load Balancer
- Azure Firewall
- A network address translation (NAT) gateway
- Network security group (NSG)
- Network policy
- User defined routes (route tables)
- Virtual network peering

Note

Azure Virtual Network Verifier (Preview) can't access any external or third-party networking resources, such as a custom firewall. If the connectivity analysis doesn't detect any blocked traffic, we recommend that you perform a manual check of any external networking to cover all hops in the traffic flow.

Currently, clusters using Azure CNI Overlay aren't supported for this feature. Support for CNI Overlay is planned for August 2025.

1. Navigate to your cluster in the Azure portal. In the sidebar, navigate to the Settings -> Node pools blade.
2. Identify the nodepool you want to run a connectivity analysis from. Click on the nodepool to select it as the scope.
3. Select "Connectivity analysis (Preview)" from the toolbar at the top of the page. If you don't see it, click on the three dots "..." in the toolbar at the top of the page to open the expanded menu.
image
4. Select a Virtual Machine Scale Set (VMSS) instance as the source. The source IP addresses are populated automatically.
5. Select a public domain name/endpoint as the destination for the analysis, one example is `mcr.microsoft.com`. The destination IP addresses are also populated automatically.
6. Run the analysis and wait up to 2 minutes for the results. In the resulting diagram, identify the associated Azure network resources and where traffic is blocked. To view the detailed analysis output, click on the "JSON output" tab or click into the arrows in the diagram.

Manual troubleshooting

If the Azure Virtual Network Verifier (Preview) tool doesn't provide enough insight into the issue, you can manually troubleshoot outbound connectivity failures by using the Secure Shell protocol (SSH) to connect to the node. To make the connection, follow the instructions in [Connect to Azure Kubernetes Service \(AKS\) cluster nodes for maintenance or troubleshooting](#).

Then, test the connectivity on the cluster by following these steps:

1. After you connect to the node, run the `nc` and `dig` commands:

Bash

```
nc -vz mcr.microsoft.com 443  
dig mcr.microsoft.com 443
```

ⓘ Note

If you can't access the node through SSH, you can test the outbound connectivity by running the [`az vmss run-command invoke`](#) command against the Virtual Machine Scale Set instance:

Azure CLI

```
# Get the VMSS instance IDs.  
az vmss list-instances --resource-group <mcr-resource-group-name> \  
--name <vmss-name> \  
--output table
```

```
# Use an instance ID to test outbound connectivity.
az vmss run-command invoke --resource-group <mc-resource-group-name> \
    --name <vmss-name> \
    --command-id RunShellScript \
    --instance-id <vmss-instance-id> \
    --output json \
    --scripts "nc -vz mcr.microsoft.com 443"
```

2. If you try to create an AKS cluster by using an HTTP proxy, run the `nc`, `curl`, and `dig` commands after you connect to the node:

Bash

```
# Test connectivity to the HTTP proxy server from the AKS node.
nc -vz <http-s-proxy-address> <port>

# Test traffic from the HTTP proxy server to HTTPS.
curl --proxy http://<http-proxy-address>:<port>/ --head
https://mcr.microsoft.com

# Test traffic from the HTTPS proxy server to HTTPS.
curl --proxy https://<https-proxy-address>:<port>/ --head
https://mcr.microsoft.com

# Test DNS functionality.
dig mcr.microsoft.com 443
```

① Note

If you can't access the node through SSH, you can test the outbound connectivity by running the `az vmss run-command invoke` command against the Virtual Machine Scale Set instance:

Azure CLI

```
# Get the VMSS instance IDs.
az vmss list-instances --resource-group <mc-resource-group-name> \
    --name <vmss-name> \
    --output table

# Use an instance ID to test connectivity from the HTTP proxy server to
# HTTPS.
az vmss run-command invoke --resource-group <mc-resource-group-name> \
    --name <vmss-name> \
    --command-id RunShellScript \
    --instance-id <vmss-instance-id> \
    --output json \
```

```

--scripts "curl --proxy http://<http-proxy-address>:<port>/ --head
https://mcr.microsoft.com"

# Use an instance ID to test connectivity from the HTTPS proxy server to
# HTTPS.
az vmss run-command invoke --resource-group <mc-resource-group-name> \
    --name <vmss-name> \
    --command-id RunShellScript \
    --instance-id <vmss-instance-id> \
    --output json \
    --scripts "curl --proxy https://<https-proxy-address>:<port>/ --head
https://mcr.microsoft.com"

# Use an instance ID to test DNS functionality.
az vmss run-command invoke --resource-group <mc-resource-group-name> \
    --name <vmss-name> \
    --command-id RunShellScript \
    --instance-id <vmss-instance-id> \
    --output json \
    --scripts "dig mcr.microsoft.com 443"

```

Solution

The following table lists specific reasons why traffic might be blocked, and the corresponding solution for each reason:

[] [Expand table](#)

Issue	Solution
Traffic is blocked by firewall rules, a proxy server, or Network Security Group (NSG)	This issue occurs when the AKS required ports or Fully Qualified Domain Names (FQDNs) are blocked by a firewall, proxy server, or NSG. Ensure these ports and FQDNs are allowed. To determine what's blocked, check the connectivity provided in the preceding Cause section. For more information about AKS required ports and FQDNs, see Outbound network and FQDN rules for Azure Kubernetes Service (AKS) clusters .
The AAAA (IPv6) record is blocked on the firewall	On your firewall, verify that nothing exists that would block the endpoint from resolving in Azure DNS.
Private cluster can't resolve internal Azure resources	In private clusters, the Azure DNS IP address (168.63.129.16) must be added as an upstream DNS server if custom DNS is used. Verify that the address is set on your DNS servers. For more information, see Create a private AKS cluster and What is IP address 168.63.129.16? .

More information

- General troubleshooting of AKS cluster creation issues

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Cluster pending operation (OperationNotAllowed) errors

Article • 04/10/2024

This article discusses how to troubleshoot OperationNotAllowed errors that occur when you try to start, upgrade, or scale a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- [Azure CLI](#)

Symptoms

You experience multiple symptoms that include one of the following error messages:

Operation is not allowed: Another operation (<operation-name>) is in progress, please wait for it to finish before starting a new operation. See <https://aka.ms/aks-pending-operation> for more details

Or, if it's an operation on an agent pool:

Operation is not allowed: Another agentpool operation (<operation-name>) is in progress, please wait for it to finish before starting a new operation. See <https://aka.ms/aks-pending-operation> for more details

Or:

Managed Cluster operation is not allowed: Another operation (<operation-name>) is in progress on agent pool (<agent-pool-name>), please wait for it to finish before starting a new operation. See <https://aka.ms/aks-pending-operation> for more details

Cause

Some operations take time to run. Those operations block other operations if they aren't finished.

Solution 1: Wait until the operation finishes

In the following example, if you start a cluster from one client while the cluster is being updated from another client, the "OperationNotAllowed" error occurs.

```
Azure CLI

az aks start -n <myAKSCluster> -g <myResourceGroup>

(OperationNotAllowed) managed cluster is in Provisioning State(Updating) and
Power State(Running), starting cannot be performed The previous operation
started at '2024-02-21T13:33:55Z' and elapsed time is: '00:00:00' (RFC3339
format)
Code: OperationNotAllowed
Message: managed cluster is in Provisioning State(Starting) and Power
State(Running), starting cannot be performed The previous operation started
at '2024-02-21T13:33:55Z' and elapsed time is: '00:00:00' (RFC3339 format)
```

To resolve such issue, you can wait until the blocking operation finishes, or try aborting the long running operation by using the [az aks operation-abort](#) command.

Solution 2: Ensure you don't perform two similar operations in a row

If you execute an operation on a cluster that's already in the desired state, the "OperationNotAllowed" error occurs.

For example, if a cluster is already stopped, executing another stop operation triggers this error:

```
Azure CLI

az aks stop -n <myAKSCluster> -g <myResourceGroup>

(OperationNotAllowed) managed cluster is not currently running, stopping
cannot be performed; The stop operation started at '2024-02-13T15:01:15Z'
and elapsed time is: '7 days and 01:16:37' (RFC3339 format)
Code: OperationNotAllowed
Message: managed cluster is not currently running, stopping cannot be
performed; The stop operation started at '2024-02-13T15:01:15Z' and elapsed
time is: '7 days and 01:16:37' (RFC3339 format)
```

To resolve such issue, start the cluster before attempting to stop it again.

Solution 3: Get the current cluster status before you try an operation

You can also determine the current status of the cluster before you try an operation. To help diagnose the issue, run the following `az aks show` command to retrieve detailed status about the cluster.

Azure CLI

```
az aks show --resource-group <myResourceGroup> --name <myAKSCluster> --output table
```

Then, use the following table to take the appropriate action based on the command results. (See the `ProvisioningState` column in the `az aks show` command output table.)

[+] Expand table

Command result	Action
Cluster is actively updating	Wait until the operation finishes.
Cluster update failed	Locate the reason for the failure in the activity logs.
Cluster update succeeded	Retry the start, scale, or other previously failed operation.

Solution 4: Retry the operation

There are scenarios where an operation fails because of a transient issue, and is left with an inconsistent state.

In the following example, a deletion was issued on the node pool `<agentpool>` but that deletion isn't completed yet. Once a deletion started, no other operation can be made on the resource. That's why the scale operation fails with the "OperationNotAllowed" error.

Output

```
{
  "code": "OperationNotAllowed",
  "details": null,
  "message": "Unable to perform 'Scaling' operation on 'agentpool' since deletion was issued on 'agentpool'. The only allowed operation is deletion once deletion has started. The delete operation started at '2024-01-09T04:29:12Z' and elapsed time is: '00:30:28' (RFC3339 format)",
```

```
"subcode": ""  
}
```

To resolve such issue, wait for the deletion to finish. If it's not finished after a few hours, retry that deletion later.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?



429 Too Many Requests errors

Article • 02/20/2025

This article discusses how to troubleshoot failures that are caused by "429 Too Many Requests" errors on your Microsoft Azure Kubernetes Service (AKS) clusters (or clusters that use another Kubernetes implementation on Azure).

Symptoms

You receive errors that resemble the following text:

Service returned an error.

Status=429

Code="OperationNotAllowed"

Message="The server rejected the request because too many requests have been received for this subscription."

```
Details=[{  
  "code": "TooManyRequests",  
  "message": {  
    "operationGroup": "\"HighCostGetVMScaleSet30Min\"",  
    "startTime": "\"2020-09-20T07:13:55.2177346+00:00\"",  
    "endTime": "\"2020-09-20T07:28:55.2177346+00:00\"",  
    "allowedRequestCount": 1800,  
    "measuredRequestCount": 2208  
  },  
  "target": "HighCostGetVMScaleSet30Min"  
}]
```

InnerError={"internalErrorCode": "TooManyRequestsReceived"}

Cause: Excessive call volumes cause Azure to throttle your subscription

A Kubernetes cluster on Azure (with or without AKS) that does a frequent scale up or scale down, or uses the cluster autoscaler, can cause a large volume of HTTP calls. This call volume can result in failure, because it exceeds the assigned quota for your Azure subscription.

For more information about these errors, see [Throttling Azure Resource Manager requests](#) and [Troubleshooting API throttling errors](#). For information about how to analyze and identify the cause of these errors and get recommendations to resolve them, see [Analyze and identify errors by using AKS Diagnose and Solve Problems](#).

Solution 1: Upgrade to a later version of Kubernetes

Run Kubernetes 1.18.x or later. These versions contain many improvements that are described in [AKS throttling/429 errors](#) and [Support large clusters without throttling](#). However, if you still see throttling (due to the actual load or number of clients in the subscription), you can try the following solutions.

Solution 2: Increase the autoscaler scan interval

If you [find the "Cluster Auto-Scaler Throttling has been detected" diagnostic reports](#) caused by the cluster autoscaler, you can try to increase the [autoscaler scan interval](#) to reduce the number of calls to virtual machine scale sets (VMSS) from the cluster autoscaler.

Solution 3: Reconfigure third-party applications to make fewer calls

When you [filter by user agents in the "View request rate and throttle details" diagnostic](#), if you find third-party applications (such as monitoring applications) that make an excessive number of GET requests, change the settings of these applications to reduce the frequency of the GET calls. In addition, make sure that the application clients use exponential backoff when calling Azure APIs.

Solution 4: Split your clusters into different subscriptions or regions

If there are numerous clusters and node pools that use virtual machine scale sets, try to split the clusters into different subscriptions or regions (within the same subscription). Most Azure API limits are shared limits at the subscription-region level. For example, all clusters and clients within sub one and the East US region share a limit for the virtual machine scale sets GET API. Hence, you can move or scale new AKS clusters in a new region and get unblocked on Azure API throttling. This technique helps if you expect the

clusters to have high activity (for example, if you have an active cluster autoscaler). It also helps if you have many clients (such as Rancher, Terraform, and so on). Since all clusters are different in their elasticity and the number of clients polling Azure APIs, there are no generic guidelines on the number of clusters that you can run per subscription-region level. For specific guidance, you can create a support ticket.

Analyze and identify errors by using AKS Diagnose and Solve Problems

For an AKS cluster, you can use [AKS Diagnose and Solve Problems](#) to analyze and identify the cause of these errors and get recommendations to resolve them. Navigate to your cluster in the Azure portal, and select **Diagnose and solve problems** in the left navigation to open AKS Diagnose and Solve Problems. Search and open *Azure Resource Request Throttling*, where you can get a report with a series of diagnostics. Those diagnostics can show if the cluster has experienced any request rate throttling (429 responses) of Azure Resource Manager (ARM) or Resource Provider (RP), and where the throttling comes from. For example:

- **Request Rate Throttling has been detected for your Cluster:** This diagnostic provides some general recommendations if throttling has been detected in the current AKS cluster.
- **Cluster Auto-Scaler Throttling has been detected:** This diagnostic shows up if throttling has been detected and originated from the cluster autoscaler.

To reduce the volume of requests from the cluster autoscaler, use the following methods:

- Increase the autoscaler scan interval to reduce the number of calls from the cluster autoscaler to virtual machine scale sets. This method may have a negative latency impact on the time taken to scale up because the cluster autoscaler waits longer before calling Azure Compute Resource Provider (CRP) for a new virtual machine.
- Make sure the cluster is on a minimum Kubernetes version of 1.18. Kubernetes version 1.18 and later versions handle request rate backoff better when 429 throttling responses are received. We highly recommend staying within supported Kubernetes versions to receive security patches.
- **Throttling - Azure Resource Manager:** This diagnostic shows the number of throttled requests in the specified time range in the AKS cluster.
- **Request Rate - Azure Resource Manager:** This diagnostic shows the total number of requests in the specified time range in the AKS cluster.

- **View request rate and throttle details:** This diagnostic has multiple diagrams to determine the throttling details, including throttled requests and total requests. You can also filter the results by using the following dimensions:
 - Host: The host where HTTP status 429 responses were detected. Azure Resource Manager throttles come from `management.azure.com`; anything else is a lower-layer resource provider.
 - User agent: Requests with a specified user agent that were throttled.
 - Operation: Operations where HTTP status 429 responses were detected.
 - Client IP: The client IP address that sent the throttled requests.

Request throttling can be caused by a combination of any cluster in this subscription, not just the request rate for this cluster.

Example 1: Cluster Auto-Scaler throttling

This example is about analyzing throttling caused by the cluster autoscaler.

If you find the **Cluster Auto-Scaler Throttling has been detected** diagnostic in AKS **Diagnose and Solve Problems > Known Issues, Availability and Performance > Azure Resource Request Throttling**, it indicates requests issued by the cluster autoscaler have been throttled.

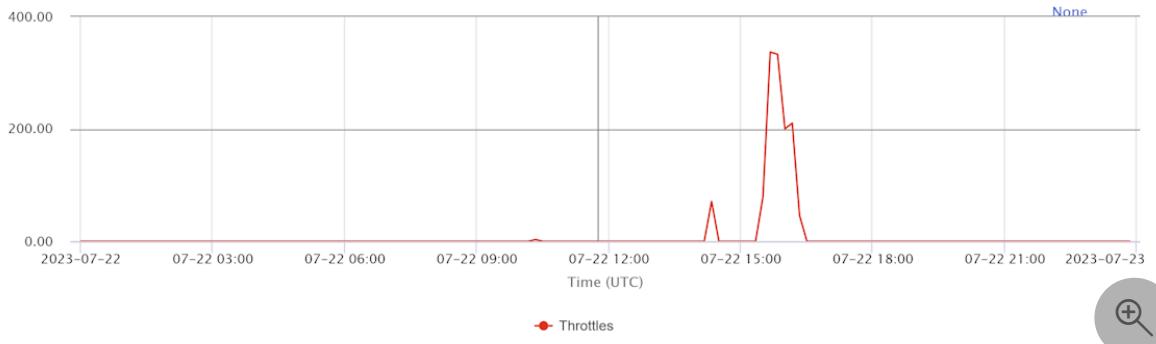
⚠️ Cluster Auto-Scaler Throttling has been detected

Summary	Throttling has been detected that originated from the cluster auto-scaler. Auto-Scale operation throttling can negatively impact the stability of your cluster. This issue can cause grow and shrink operation failures for node pool scaling, disk attachment failures, node provisioning failures, as well as any other general operational call to the Compute Resource Provider, which may leave you unable to manage your Compute resources, or your cluster.
Recommended Action	Reduce the volume of calls emanating from the cluster auto-scalers in this subscription: Show tasks from auto-scaler . These are some actions to take to try and reduce the volume of calls from the cluster auto-scaler. <ul style="list-style-type: none"> • <i>Increase the auto-scaler scan interval</i> - Increasing the auto-scaler scan interval will reduce the number of calls to VMSS for each auto-scaling cluster. • <i>Make sure clusters are on minimum K8s version 1.18</i> - Kubernetes version 1.18 and later better handles request rate back-off when 429 responses are received.

You can find the number of throttled requests and when the requests are throttled in the **Throttling - Azure Resource Manager** diagnostic.

Throttling - Azure Resource Manager

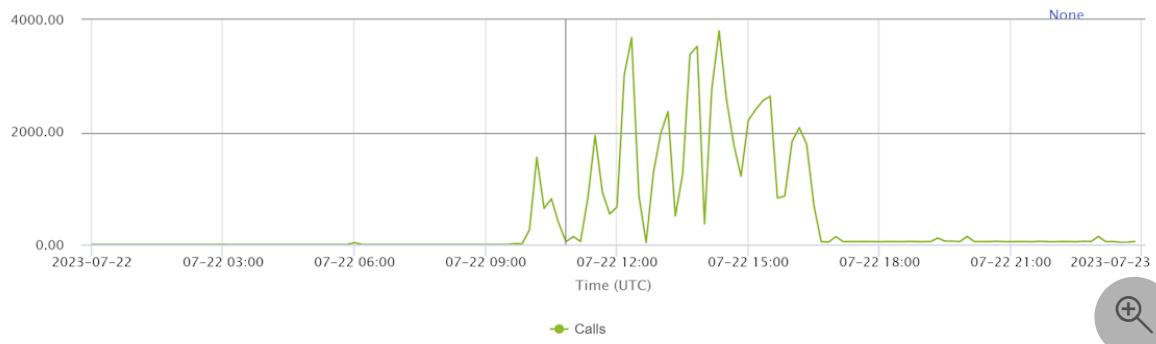
Number of throttled requests per 10 minutes for this cluster. Max throttled count was 337 throttles in a single 10 minute window.



You can find the number of all ARM requests in the same time period.

Request Rate - Azure Resource Manager

Total request rate per 10 minutes for this cluster. Note that request throttling can be caused by a combination of any cluster in this subscription, not just the request rate for this cluster.



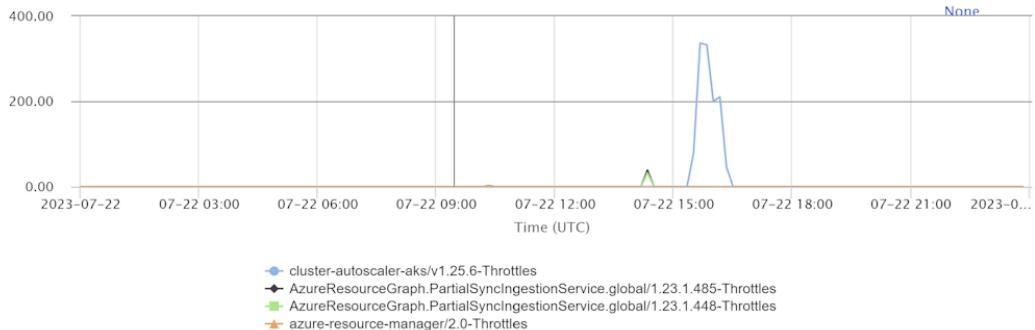
You can check the [View request rate and throttle details](#) diagnostic to find the throttling details. Select **429s by User Agent** from the **Select filter** drop-down list, and you can see that autoscaler requests are throttled from 15:00 to 16:00.

View request rate and throttle details

Select filter 429s by User Agent ▾

Throttles by User Agent

User Agents where HTTP status 429 responses were detected. Note that request throttling can be caused by a combination of any cluster in this subscription, not just the request rate for this cluster.



You can also find the total number of throttled requests for the cluster autoscaler and other user agents.

Total Throttles by User Agent

Found a total of 4 User Agents that had throttled requests. Note that request throttling can be caused by a combination of any cluster in this subscription, not just the request rate for this cluster.

User Agent	Calls	Throttles
Go/go1.19.7 (amd64-linux) go-autorest/v14.2.1 cluster-autoscaler-aks/v1.25.6	16468	1205
AzureResourceGraph.PartialSyncIngestionService.global/1.23.1.485	19722	39
AzureResourceGraph.PartialSyncIngestionService.global/1.23.1.448	10808	32
azure-resource-manager/2.0	68	3

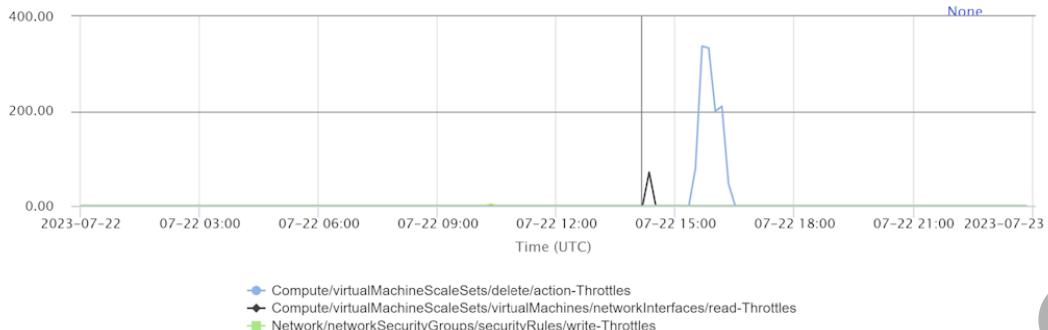
You can also filter throttles by operations. VMSS VM delete operation is throttled in this case.

View request rate and throttle details

Select filter 429s by Operation ▾

Throttles by Operation

Operations where HTTP status 429 responses were detected. Note that request throttling can be caused by a combination of any cluster in this subscription, not just the request rate for this cluster.



You can find the number of throttled requests and all requests grouped by operations.

Total Throttles by Operation

Found a total of 3 Operations that had throttled requests. Note that request throttling can be caused by a combination of any cluster in this subscription, not just the request rate for this cluster.

Operation	Calls	Throttles
Compute/virtualMachineScaleSets/delete/action	5394	1205
Compute/virtualMachineScaleSets/virtualMachines/networkInterfaces/read	10345	71
Network/networkSecurityGroups/securityRules/write	11	3



Then, you can follow the suggestions in the **Recommended Action** to reduce the throttles.

⚠ Cluster Auto-Scaler Throttling has been detected

Summary

Throttling has been detected that originated from the cluster auto-scaler. Auto-Scale operation throttling can negatively impact the stability of your cluster. This issue can cause grow and shrink operation failures for node pool scaling, disk attachment failures, node provisioning failures, as well as any other general operational call to the Compute Resource Provider, which may leave you unable to manage your Compute resources, or your cluster.

Recommended Action

Reduce the volume of calls emanating from the cluster auto-scalers in this subscription: [Solve auto-scaler throttling issues](#). These are some actions to take to try and reduce the volume of calls from the cluster auto-scaler.

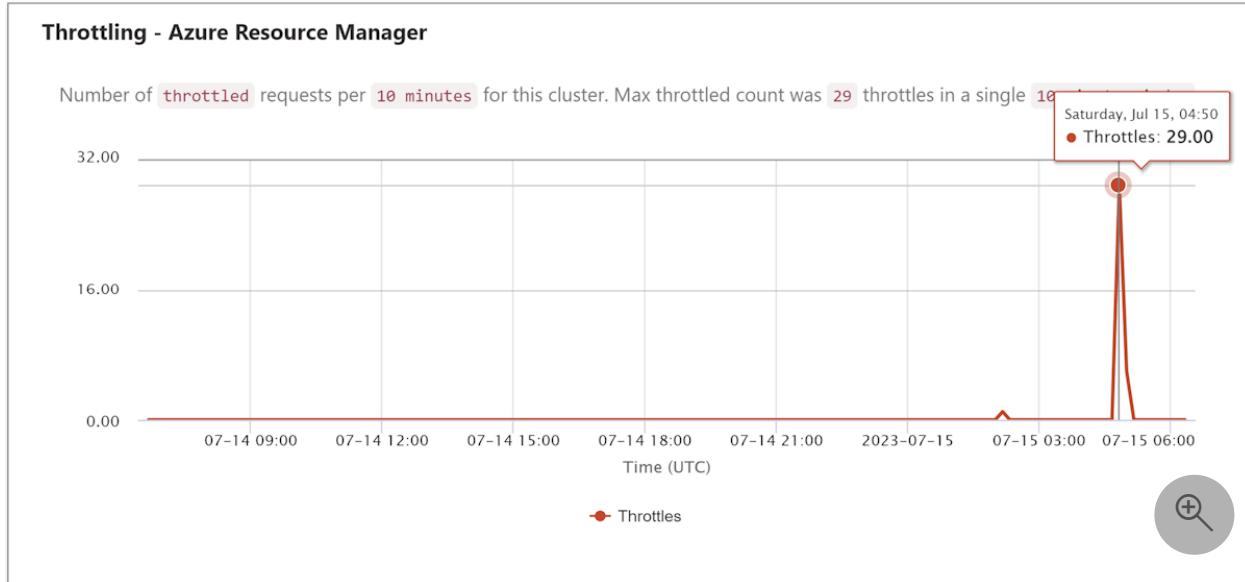
- *Increase the auto-scaler scan interval* - Increasing the auto-scaler scan interval will reduce the number of calls to VMSS for each auto-scaling cluster.
- *Make sure clusters are on minimum K8s version 1.18* - Kubernetes version 1.18 and later better handles request rate back-off when 429 responses are received.



Example 2: Cloud Provider throttling

This example is about the throttles caused by the Cloud Provider. It often happens when operating resources in larger clusters, for example, provisioning an Azure Load Balancer in a cluster that has more than 500 nodes.

If you find throttling in your cluster, you can see the throttling details in the **View request rate and throttle details** diagnostic. Select **429s by User Agent** from the **Select filter** drop-down list, and you can see that cloud provider requests were throttled from 03:00 to 06:00.



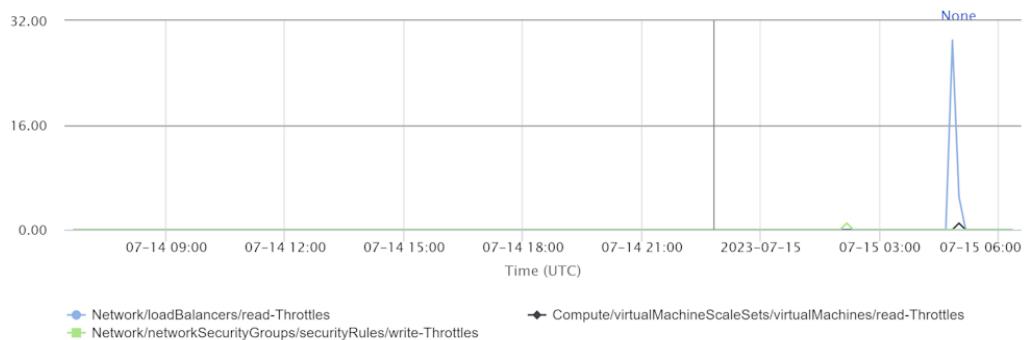
You can also filter by operations to find out that the throttled operation is "Network/loadBalancers/read".

View request rate and throttle details

Select filter 429s by Operation ▾

Throttles by Operation

Operations where HTTP status 429 responses were detected. Note that request throttling can be caused by a combination of any cluster in this subscription, not just the request rate for this cluster.



You can use the AKS preview feature [Node IP-based Load Balancer](#) to reduce this throttle.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

Yes

No

Provide product feedback

Azure Kubernetes Service (AKS) common issues FAQ

FAQ

This article answers frequently asked questions (FAQ) about common issues that can occur when you're working with an Azure Kubernetes Service (AKS) cluster.

In general, where do I find information about debugging Kubernetes problems?

Try the [official guide to troubleshooting Kubernetes clusters](#). There's also a [troubleshooting guide](#) that was published by a Microsoft engineer. This guide helps you troubleshoot pods, nodes, clusters, and other features.

I'm trying to enable Kubernetes role-based access control (Kubernetes RBAC) on an existing cluster. How can I do that?

Enabling Kubernetes role-based access control (Kubernetes RBAC) on existing clusters isn't supported at this time. This feature can be set only when you create new clusters. Kubernetes RBAC is enabled by default when you use the [Azure CLI](#), the [Azure portal](#), or an API version later than `2020-03-01`.

Can I move my cluster to a different subscription, or move my subscription with my cluster to a new tenant?

No. If you've moved your AKS cluster to a different subscription or the cluster's subscription to a new tenant, the cluster won't function because of missing cluster identity permissions. AKS doesn't support moving clusters across subscriptions or tenants because of this constraint. For more information, see [Operations FAQ](#).

What naming restrictions are enforced for AKS resources and parameters?

Naming restrictions are implemented by both the Azure platform and AKS. If a resource name or parameter breaks one of these restrictions, an error is returned that asks you provide a different input. The following common naming guidelines apply:

- Cluster names must be 1-63 characters in length. The only allowed characters are letters, numbers, dashes, and underscore. The first and last character must be a letter or a number.
- The AKS node or *MC_* resource group name combines the resource group name and resource name. The autogenerated syntax of *MC_resourceGroupName_resourceName_AzureRegion* must be no greater than 80 characters in length. If necessary, reduce the length of your resource group name or AKS cluster name. You may also [customize your node's resource group name](#).
- The Domain Name System (DNS) prefix must start and end with alphanumeric values and must be between 1-54 characters in length. Valid characters include alphanumeric values and hyphens (""). The DNS prefix can't include special characters, such as periods (".").
- AKS node pool names must be all lowercase. The names must be 1-12 characters in length for Linux node pools and 1-6 characters for Windows node pools. A name must start with a letter, and the only allowed characters are letters and numbers.
- The *admin-username*, which sets the administrator user name for Linux nodes, must start with a letter. This user name may only contain letters, numbers, hyphens, and underscores. It has a maximum length of 32 characters.
- The *aksmanagedap* is blocked as a reserved name for AKS system components. It can't be used for nodepools.

For more information about naming convention. see the following resources:

- [Naming rules and restrictions for Azure resources](#)
- [Abbreviation recommendations for Azure resources](#)

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Application failures caused by the "argument list too long" error message

Article • 10/23/2024

This article discusses troubleshooting strategies for resolving application failures that are caused by the "argument list too long" error message in Microsoft Azure Kubernetes Service (AKS).

Symptoms

Your application fails when the [kubelet](#) tries to run the executable, and you receive an error message that resembles the following output:

```
| standard_init_linux.go:228: exec user process caused: argument list too long
```

Cause 1: The argument list provided to the executable is too long

The arguments that are provided to the application's executable are too long to be processed.

Solution: Shorten the argument list

Eliminate any redundant or unnecessary arguments that you specify for the executable.

Cause 2: The set of environment variables provided to the executable is too large

If too many services are deployed in one namespace, the environment variable list can become too large, and the kubelet will produce the error message when it tries to run the executable. The error occurs because the kubelet adds environment variables that record the host and port for each active service, so that [services can use this information to find other active services](#).

Solution 1: Reduce the number of services that are active

You can reduce the total number of active services, so that the kubelet adds a smaller number of overall environment variables.

Solution 2: Reconfigure the kubelet so that it doesn't add environment variables for the service host and port

Within the [PodSpec core API](#), set the `enableServiceLinks` field to `false`. This change reconfigures the kubelet behavior so that the host and port aren't automatically added as environment variables for each active service.

Warning

If your service relies on these environment variables to find other services, this field change will cause the service to fail. To avoid this scenario, rely on DNS for service discovery instead of environment variables, by using [CoreDNS](#).

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?



[Provide product feedback](#)

"Changing property 'imageReference' is not allowed" error message while upgrading or scaling an AKS cluster

Article • 10/17/2024

Symptoms

When you try to upgrade or scale a Microsoft Azure Kubernetes Service (AKS) cluster, you might receive the "Changing property 'imageReference' is not allowed" error message.

Cause

Unexpected results might occur if you modify or delete tags and other properties of resources within the *MC_** resource group. Also, altering the resources within the *MC_** group in the AKS cluster will break the service-level objective (SLO).

Solution

Add another node pool, and then delete the affected node pool.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

"CreateOrUpdateVirtualNetworkLinkFailed" error when updating or upgrading an AKS cluster

Article • 04/10/2024

This article provides a solution to the "CreateOrUpdateVirtualNetworkLinkFailed" error code that occurs when you try to update or upgrade a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

[Azure CLI](#)

Symptoms

An AKS cluster update or upgrade operation fails and returns the following error message:

Code: CreateOrUpdateVirtualNetworkLinkFailed - SubCode: BadRequest

Message: Reconcile private dns failed

Details: Create or update virtual network link failed. Subscription: <SubscriptionID>; resource group: <RGName>; private dns zone: <GUID>.privatelink.<region>.azmk8s.io; virtual network link: <VNET_Link>.

Message: A virtual network cannot be linked to multiple zones with overlapping namespaces. You tried to link virtual network with '<GUID>.privatelink.<region>.azmk8s.io' and '<GUID>.privatelink.<region>.azmk8s.io' zones.

Cause

This error happens in this scenario:

- You disassociate the original private Domain Name System (DNS) zone of the AKS cluster.
- You link a private DNS zone that has the same name as the original zone but is located in a different resource group or subscription.

That's why you see the same private DNS zone name "<GUID>.privatelink.<region>.azmk8s.io" in the error message. The first is the new zone in the new resource group or subscription, while the second is the original zone created with the AKS cluster.

Solution

To resolve this issue, follow these steps:

1. Remove the link between the AKS cluster's virtual network (VNET) and the private DNS zone created in the wrong resource group or subscription.
2. Update the cluster by running the following command:

Azure CLI

```
az aks update -n <myAKSCluster> -g <myResourceGroup>
```

The command output should show the cluster's `ProvisioningState` as `Running`.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Increased memory usage reported in Kubernetes 1.25 or later versions

Article • 03/03/2025

This article discusses how to resolve an *increased reported memory usage* issue in Microsoft Azure Kubernetes 1.25 or a later version.

Symptoms

You experience one or more of the following symptoms:

- Pods report increased memory usage after you upgrade a Microsoft Azure Kubernetes Service (AKS) cluster to Kubernetes 1.25 or a later version.
- A node reports memory usage that's greater than in earlier versions of Kubernetes when you run the [kubectl top node](#) command.
- Increased pod evictions and memory pressure occur within a node.

Cause

This increase is caused by a change in memory accounting within version 2 of the Linux control group (`cgroup`) API. [Cgroup v2](#) is now the default `cgroup` version for Kubernetes 1.25 on AKS.

Note

This issue is distinct from the memory saturation in nodes that's caused by applications or frameworks that aren't aware of `cgroup` v2. For more information, see [Memory saturation occurs in pods after cluster upgrade to Kubernetes 1.25](#).

Solution

- If you observe frequent memory pressure on the nodes, upgrade your subscription to increase the amount of memory that's available to your virtual machines (VMs).
- If you see a higher eviction rate on the pods, [use higher limits and requests for pods](#).

- `cgroup` v2 uses a different API than `cgroup` v1. If there are any applications that directly access the `cgroup` file system, update them to later versions that support `cgroup` v2. For example:
 - **Third-party monitoring and security agents:**

Some monitoring and security agents depend on the `cgroup` file system. Update these agents to versions that support `cgroup` v2.
 - **Java applications:**

Use versions that fully support `cgroup` v2:

 - OpenJDK/HotSpot: `jdk8u372`, `11.0.16`, `15`, and later versions.
 - IBM Semeru Runtimes: `8.0.382.0`, `11.0.20.0`, `17.0.8.0`, and later versions.
 - IBM Java: `8.0.8.6` and later versions.
 - **uber-go/automaxprocs:**

If you're using the `uber-go/automaxprocs` package, ensure the version is `v1.5.1` or later.
- An alternative temporary solution is to revert the `cgroup` version on your nodes by using the DaemonSet. For more information, see [Revert to cgroup v1 DaemonSet ↗](#).

ⓘ Important

- Use the DaemonSet cautiously. Test it in a lower environment before applying to production to ensure compatibility and prevent disruptions.
- By default, the DaemonSet applies to all nodes in the cluster and reboots them to implement the `cgroup` change.
- To control how the DaemonSet is applied, configure a `nodeSelector` to target specific nodes.

ⓘ Note

If you experience only an increase in memory use without any of the other symptoms that are mentioned in the "Symptoms" section, you don't have to take any action.

Status

We're actively working with the Kubernetes community to resolve the underlying issue. Progress on this effort can be tracked at [Azure/AKS Issue #3443](#).

As part of the resolution, we plan to adjust the eviction thresholds or update [resource reservations](#), depending on the outcome of the fix.

Reference

- Node memory usage on cgroupv2 reported higher than cgroupv1 ([GitHub issue](#))

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

Missing or invalid service principal when creating an AKS cluster

Article • 04/30/2025

This article discusses how to troubleshoot a service principal that isn't found or is invalid when you try to create a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

Use Azure CLI version 2.0.81 or later for running the commands in this article.

Cause

When you create an AKS cluster, AKS requires a service principal or managed identity to manage resources on your behalf. By default, AKS uses a system-assigned managed identity. If you prefer to use a service principal instead, be aware that AKS does not automatically create one for you. You'll have to provide your own service principal and reference it during cluster creation. For more information, see [Use a service principal with Azure Kubernetes Service \(AKS\)](#).

Although service principals are supported for AKS authentication, we recommend using a system-assigned managed identity. This identity can simplify credential management and is the default option for new clusters.

Additionally, when you create a service principal, make sure that it's propagated across all regions by Microsoft Entra ID. If this propagation takes too long, the cluster might fail validation because AKS can't locate the service principal.

Solution

Make sure that there's a valid, findable service principal. To do this, use one of the following methods:

- When you create an AKS cluster, consider using an existing service principal that has already propagated across regions. Although there's no direct way to verify the propagation status, you can verify functionality by using a previously deployed service principal. Alternatively, if you're using a new principal, allow 5-10 minutes for the principal to propagate before you start the cluster creation.
- To verify that the service principal is ready, execute the `az ad sp show --id <appId>` command and check the output before proceeding with the creation of the AKS cluster.

- If you use automation scripts, add time delays between service principal creation and AKS cluster creation. We recommend a delay of 5 to 10 minutes.
- If you use the [Azure portal](#), return to the cluster settings after you try to create the cluster, and then retry the validation page after a few minutes.

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Pod is stuck in CrashLoopBackOff mode

Article • 04/07/2025

If a pod has a `CrashLoopBackOff` status, then the pod probably failed or exited unexpectedly, and the log contains an exit code that isn't zero. Here are several possible reasons why your pod is stuck in `CrashLoopBackOff` mode:

- 1. Application failure:** The application inside the container crashes shortly after starting, often due to misconfigurations, missing dependencies, or incorrect environment variables.
- 2. Incorrect resource limits:** If the pod exceeds its CPU or memory resource limits, Kubernetes might kill the container. This issue can happen if resource requests or limits are set too low.
- 3. Missing or misconfigured ConfigMaps/Secrets:** If the application relies on configuration files or environment variables stored in ConfigMaps or Secrets but they're missing or misconfigured, the application might crash.
- 4. Image pull issues:** If there's an issue with the image (for example, it's corrupted or has an incorrect tag), the container might not start properly and fail repeatedly.
- 5. Init containers failing:** If the pod has init containers and one or more fail to run properly, the pod will restart.
- 6. Liveness/Readiness probe failures:** If liveness or readiness probes are misconfigured, Kubernetes might detect the container as unhealthy and restart it.
- 7. Application dependencies not ready:** The application might depend on services that aren't yet ready, such as databases, message queues, or other APIs.
- 8. Networking issues:** Network misconfigurations can prevent the application from communicating with necessary services, causing it to fail.
- 9. Invalid commands or arguments:** The container might be started with an invalid `ENTRYPOINT`, command, or argument, leading to a crash.

For more information about the container status, see [Pod Lifecycle - Container states](#).

Consider the following options and their associated [kubectl](#) commands.

[] Expand table

Option	kubectl command
Debug the pod itself	<code>kubectl describe pod <pod-name></code>
Debug the replication controllers	<code>kubectl describe replicationcontroller <controller-name></code>
Read the termination message	<code>kubectl get pod <pod-name> --output=yaml</code>

Option	kubectl command
Examine the logs ↗	<code>kubectl logs <pod-name></code>

⚠ Note

A pod can also have a `CrashLoopBackOff` status if it has finished deployment, but it's configured to keep restarting even if the exit code is zero. For example, if you deploy a busybox image without specifying any arguments, the image starts, runs, finishes, and then restarts in a loop:

Console

```
$ kubectl run nginx --image nginx
pod/nginx created

$ kubectl run busybox --image busybox
pod/busybox created

$ kubectl get pods --watch
NAME      READY   STATUS            RESTARTS   AGE
busybox   0/1    ContainerCreating  0          3s
nginx     1/1    Running           0          11s
busybox   0/1    Completed         0          3s
busybox   0/1    Completed         1          4s
busybox   0/1    CrashLoopBackOff  1          5s

$ kubectl describe pod busybox
Name:           busybox
Namespace:      default
Priority:       0
Node:          aks-nodepool<number>-<resource-group-hash-number>-
vmss<number>/<ip-address-1>
Start Time:    Wed, 16 Aug 2023 09:56:19 +0000
Labels:         run=busybox
Annotations:   <none>
Status:        Running
IP:            <ip-address-2>
IPS:
  IP:  <ip-address-2>
Containers:
  busybox:
    Container ID:  containerd://<64-digit-hexadecimal-value-1>
    Image:         busybox
    Image ID:      docker.io/library/busybox@sha256:<64-digit-hexadecimal-
value-2>
    Port:          <none>
    Host Port:    <none>
    State:         Waiting
    Reason:        CrashLoopBackOff
    Last State:   Terminated
```

Reason:	Completed
Exit Code:	0
Started:	Wed, 16 Aug 2023 09:56:37 +0000
Finished:	Wed, 16 Aug 2023 09:56:37 +0000
Ready:	False
Restart Count:	2

If you don't recognize the issue after you create more pods on the node, then run a pod on a single node to determine how many resources the pod actually uses.

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

"TLS: client offered only unsupported versions" error on client when connecting to the AKS API server

Article • 10/14/2024

Symptoms

When you try to connect to the Microsoft Azure Kubernetes Service (AKS) API server, you receive the following error message from your client computer or virtual machine:

tls: client offered only unsupported versions

Cause

Your version of the Transport Layer Security (TLS) protocol is out of date. The minimum supported version in AKS is TLS 1.2.

Solution

Upgrade your TLS version to 1.2. For upgrade instructions, see [Enable support for TLS 1.2 in your environment](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?



Troubleshoot common issues for Azure Linux Container Host for AKS

Article • 04/02/2025

This article provides troubleshooting steps for some of the commonly reported issues that you might experience when you use Azure Linux container hosts in Azure Kubernetes Service (AKS). For more information about how to get started using Azure Linux container hosts in AKS, see [Use Azure Linux with AKS](#).

Before you begin

Read the [official guide for troubleshooting Kubernetes clusters](#). Also, read the [Microsoft engineer's guide to Kubernetes troubleshooting](#). This guide contains commands for troubleshooting pods, nodes, clusters, and other features.

Finally, review the [list of known limitations in Azure Linux](#). An issue that you're trying to resolve might be one that we're already working on.

Prerequisites

- [Azure CLI](#), version 2.31 or a later version. If Azure CLI is already installed, you can find the version number by running `az --version`.

About Azure Linux Container Host for AKS

Azure Linux is an open-source Linux distribution that Microsoft created. As a lightweight OS, Azure Linux has the following features:

- Contains only the packages that are needed to run container workloads
- Undergoes Azure validation tests
- Is compatible with Azure agents

Azure Linux Container Host for AKS is an operating system image for AKS that's optimized for running container workloads. It's maintained by Microsoft and based on Azure Linux. It provides reliability and consistency from cloud to edge across the [AKS](#), [AKS on Azure Stack HCI](#), and [Azure Arc](#) products. You can use Azure Linux container hosts to do the following processes:

- Deploy Azure Linux node pools in a new cluster.

- Add Azure Linux node pools to your existing Ubuntu clusters.
- Migrate your Ubuntu nodes to Azure Linux nodes.

For more information about Azure Linux, see the [Azure Linux GitHub repository](#).

Troubleshooting checklist

Step 1: Review equivalent commands in Ubuntu and Azure Linux

Most commands in the Azure Linux OS, such as the process status (`ps`) command, resemble commands that are used in Ubuntu. However, package management is done by using the Tiny DNF (`tdnf`) command. The following table lists some common commands in Ubuntu and their equivalents in Azure Linux.

[\[+\] Expand table](#)

Ubuntu command	Suggested Azure Linux command
<code>apt -- list installed</code>	<code>rpm -qa</code>
<code>apt autoclean</code>	<code>tdnf clean all</code>
<code>apt autoremove</code>	<code>dnf autoremove</code>
<code>apt dist-upgrade</code>	<code>dnf distro-sync</code>
<code>apt download</code>	<code>tdnf download</code>
<code>apt install</code>	<code>tdnf install</code>
<code>apt install --reinstall</code>	<code>tdnf reinstall</code>
<code>apt list - upgradable</code>	<code>dnf list updates</code>
<code>apt remove</code>	<code>tdnf remove</code>
<code>apt search</code>	<code>tdnf search</code>
<code>apt show</code>	<code>tdnf list</code>
<code>apt upgrade</code>	<code>tdnf upgrade</code>
<code>apt cache dump</code>	<code>tdnf list available</code>
<code>apt-cache dumpavail</code>	<code>tdnf list available</code>

Ubuntu command	Suggested Azure Linux command
<code>apt-cache policy</code>	<code>tdnf list</code>
<code>apt-cache rdepends</code>	<code>dnf repoquery -- alldeps - whatrequires</code>
<code>apt-cache search</code>	<code>tdnf search</code>
<code>apt-cache show</code>	<code>tdnf info</code>
<code>apt-cache stats</code>	(no exact equivalent; read the <i>Packages</i> file in the <i>/var/lib/rpm</i> folder)
<code>apt-config shell</code>	<code>dnf shell</code>
<code>apt-file list</code>	<code>dnf repoquery -l</code>
<code>apt-file search</code>	<code>tdnf provides</code>
<code>apt-get autoremove</code>	<code>dnf autoremove</code>
<code>apt-get install</code>	<code>tdnf install</code>
<code>apt-get remove</code>	<code>tdnf remove</code>
<code>apt-get update</code>	<code>dnf clean expire-cache dnf check-update</code>
<code>apt-mark auto</code>	<code>tdnf install dnf mark remove</code>
<code>apt-mark manual</code>	<code>dnf mark install</code>
<code>apt-mark showmanual</code>	<code>dnf history userinstalled</code>
<code>add-apt-repository</code>	Edit <code>/etc/yum.repos.d/*.repo</code> files
<code>apt-key add</code>	<code>rpm --import</code>

Step 2: Check the Azure Linux version

Make sure that you're using the correct version of Azure Linux. The supported version of Azure Linux for consumption is Azure Linux 2.0. In the output of the following `az aks nodepool list` command, the `osSKU` property should read `AzureLinux`.

Azure CLI

```
az aks nodepool list --resource-group <resource-group-name> --cluster-name
<aks-cluster-name>
```

Although this command might not address the issue that you're experiencing, versioning is a common issue for users who report that agents or extensions aren't

working correctly on Azure Linux.

Step 3: Understand the difference in certificate file paths

Azure Linux (and other RPM distributions) store certificates differently from Ubuntu.

On Azure Linux, the `/etc/ssl/certs` path is a symbolic link to `/etc/pki/tls/certs`. If a container expects to map `/etc/ssl/certs` to use the `ca-certificates.crt` certificate file on Azure Linux, the container instead gets a symbolic link that points to nowhere. This behavior causes certificate-related errors in the container. The container also has to map `/etc/pki` so that the container can follow the symbolic link chain. If the container has to work on both Ubuntu and Azure Linux hosts, you can map `/etc/pki` by using the `DirectoryOrCreate` type in a [hostPath volume](#).

Step 4: Update Azure CLI and the AKS preview extension

If you try to deploy an Azure Linux AKS cluster by using Azure CLI, you might receive an error message that states that the `AzureLinux` option isn't supported for the `osssku` parameter. This message means that you might be using an outdated version of the Azure CLI or the AKS preview extension. To fix this issue, take one or both of the following two actions:

- If Azure CLI isn't up to date, install the latest version. To upgrade Azure CLI, run the following [az upgrade](#) command:

```
Azure CLI
```

```
az upgrade
```

- If you have an older version of the `aks-preview` extension installed, install a newer version so that the `osssku` parameter has a value of `AzureLinux`. To upgrade the extension, run the following [az extension update](#) command:

```
Azure CLI
```

```
az extension update --name aks-preview
```

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

Troubleshoot the "CustomPrivateDNSZoneMissingPermissionError" error code

Article • 05/27/2025

This article discusses how to identify and resolve the "CustomPrivateDNSZoneMissingPermissionError" error code that occurs when you try to create or update a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- [Azure CLI](#), version 2.53.0 or a later version. To find your installed version, run `az --version`.

Symptoms

An AKS cluster create or update operation fails and returns the following error message:

Code: CustomPrivateDNSZoneMissingPermissionError

Message: Service principal or user-assigned identity must be given permission to read and write to custom private dns zone <custom-private-dns-zone-resource-id>. Check access result not allowed for action Microsoft.Network/privateDnsZones/read.

Cause

Before AKS runs a cluster create or update operation for a private cluster that uses a [custom private DNS zone](#), it checks whether the cluster's managed identity or service principal has the required permissions to control the private DNS zone. If AKS can't find the necessary permissions in cases like the following ones, it blocks the operation so that the cluster doesn't enter a failed state:

- The managed identity or service principal has been deleted.
- The managed identity or service principal has been re-created with the same name.
- An incorrect managed identity is passed.

Solution

To create the missing role assignment, follow these steps:

1. Get the resource ID of the cluster's private DNS zone by running the `az aks show` command, and store it as the `CUSTOM_PRIVATE_DNS_ZONE_ID` variable:

```
Azure CLI
```

```
CUSTOM_PRIVATE_DNS_ZONE_ID=$(az aks show \
--resource-group <aks-resource-group> \
--name <aks-cluster-name> \
--query apiServerAccessProfile.privateDnsZone \
--output tsv)
```

ⓘ Note

Because the resource ID of the custom private DNS zone was also shown in the original error message, you can alternatively assign that resource ID to the variable instead of running the `az aks show` command.

2. Assign the [Private DNS Zone Contributor](#) role to the cluster's managed identity or service principal by running the `az role assignment create` command:

```
Azure CLI
```

```
az role assignment create --role "Private DNS Zone Contributor" \
--scope $CUSTOM_PRIVATE_DNS_ZONE_ID \
--assignee <control-plane-principal-id>
```

ⓘ Note

It can take up to 60 minutes to finish granting permissions to your cluster's managed identity or service principal.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

"InvalidLoadBalancerProfileAllocatedOutboundPorts" error when creating or updating an AKS cluster

Article • 05/06/2024

This article discusses how to identify and resolve the "InvalidLoadBalancerProfileAllocatedOutboundPorts" error code that occurs when you try to create or update a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

Azure CLI, version 2.53.0 or a later version. To find your installed version, run the `az --version` command.

Symptoms

An AKS cluster create or update operation fails and returns the following error message:

Code: InvalidLoadBalancerProfileAllocatedOutboundPorts
Message: Load balancer profile allocated ports 8000 is not in an allowable range given the number of nodes and IPs provisioned. Total node count 9 requires 72000 ports but only 64000 ports are available given 1 outbound public IPs. Refer to <https://aka.ms/aks/slb-ports> for more details.

Cause

In clusters with outbound type `LoadBalancer`, the traffic originating from cluster nodes (including traffic from applications running in pods) egresses via the AKS-managed load balancer, which relies on Source Network Address Translation (SNAT) ports to establish these outbound connections.

By default, each cluster is assigned one outbound frontend IP address that allows for a total of 64,000 SNAT ports, and each worker node in the cluster is allocated a share of these ports. For example, if a cluster has 50 nodes or fewer, each worker node is allocated 1,024 ports. In some scenarios where applications running in a cluster require establishing large numbers of outbound connections, the default number of available SNAT ports may be insufficient, which leads to SNAT port exhaustion.

To solve the SNAT port exhaustion issue, use one or both of the following methods:

- Change the number of outbound frontend IP addresses. Each IP address provides an additional 64,000 SNAT ports.
- Change the number of SNAT ports assigned to each worker node.

The "InvalidLoadBalancerProfileAllocatedOutboundPorts" error occurs when the specified node count, the number of outbound frontend IP addresses, and the number of allocated ports per node don't constitute a feasible configuration.

Solution

To resolve the "InvalidLoadBalancerProfileAllocatedOutboundPorts" error, follow these steps:

1. Use the following formula to check if the desired configuration is feasible:

code

```
64,000 ports per IP / <outbound ports per node> * <number of outbound IPs> = <maximum number of nodes in the cluster>
```

ⓘ Note

When performing this check, make sure you consider node surges that happen during cluster upgrades and other operations. AKS defaults to one buffer node for upgrade operations, but this number can be modified using the [maxSurge](#) parameter.

2. Change the cluster's node count, the number of outbound frontend IP addresses, or the number of SNAT ports per node.

For more information about how to configure the allocated outbound ports and examples and calculations of the number of ports you might need, see [Configure the allocated outbound ports](#).

Reference

[Use Source Network Address Translation \(SNAT\) for outbound connections](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Create an unmanaged ingress controller

Article • 03/10/2025

An ingress controller is a piece of software that provides reverse proxy, configurable traffic routing, and TLS termination for Kubernetes services. Kubernetes ingress resources are used to configure the ingress rules and routes for individual Kubernetes services. When you use an ingress controller and ingress rules, a single IP address can be used to route traffic to multiple services in a Kubernetes cluster.

This article shows you how to deploy the [NGINX ingress controller](#) in an Azure Kubernetes Service (AKS) cluster. Two applications are then run in the AKS cluster, each of which is accessible over the single IP address.

Important

The Application routing add-on is recommended for ingress in AKS. For more information, see [Managed nginx Ingress with the application routing add-on](#).

Note

There are two open source ingress controllers for Kubernetes based on Nginx: one is maintained by the Kubernetes community ([kubernetes/ingress-nginx](#)), and one is maintained by NGINX, Inc. ([nginxinc/kubernetes-ingress](#)). This article will be using the Kubernetes community ingress controller.

Before you begin

- This article uses Helm 3 to install the NGINX ingress controller on a [supported version of Kubernetes](#). Make sure that you're using the latest release of Helm and have access to the *ingress-nginx* Helm repository. The steps outlined in this article may not be compatible with previous versions of the Helm chart, NGINX ingress controller, or Kubernetes.
- This article assumes you have an existing AKS cluster with an integrated Azure Container Registry (ACR). For more information on creating an AKS cluster with an integrated ACR, see [Authenticate with Azure Container Registry from Azure Kubernetes Service](#).
- The Kubernetes API health endpoint, `healthz` was deprecated in Kubernetes v1.16. You can replace this endpoint with the `liveness` and `readiness` endpoints instead. See

Kubernetes API endpoints for health [↗](#) to determine which endpoint to use for your scenario.

- If you're using Azure CLI, this article requires that you're running the Azure CLI version 2.0.64 or later. Run `az --version` to find the version. If you need to install or upgrade, see [Install Azure CLI][azure-cli-install].
- If you're using Azure PowerShell, this article requires that you're running Azure PowerShell version 5.9.0 or later. Run `Get-InstalledModule -Name Az` to find the version. If you need to install or upgrade, see [Install Azure PowerShell](#).

Basic configuration

To create a basic NGINX ingress controller without customizing the defaults, you'll use Helm. The following configuration uses the default configuration for simplicity. You can add parameters for customizing the deployment, like `--set controller.replicaCount=3`.

ⓘ Note

If you would like to enable [client source IP preservation](#) for requests to containers in your cluster, add `--set controller.service.externalTrafficPolicy=Local` to the Helm install command. The client source IP is stored in the request header under *X-Forwarded-For*. When you're using an ingress controller with client source IP preservation enabled, TLS pass-through won't work.

Azure CLI

Console

```
NAMESPACE=ingress-basic

helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
helm repo update

helm install ingress-nginx ingress-nginx/ingress-nginx \
--create-namespace \
--namespace $NAMESPACE \
--set
controller.service.annotations."service\.beta\.kubernetes\.io/azure-
load-balancer-health-probe-request-path"="/healthz" \
--set controller.service.externalTrafficPolicy=Local
```

ⓘ Note

In this tutorial, `service.beta.kubernetes.io/azure-load-balancer-health-probe-request-path` is being set to `/healthz`. This means if the response code of the requests to `/healthz` is not `200`, the entire ingress controller will be down. You can modify the value to other URI in your own scenario. You cannot delete this part or unset the value, or the ingress controller will still be down. The package `ingress-nginx` used in this tutorial, which is provided by [Kubernetes official](#), will always return `200` response code if requesting `/healthz`, as it is designed as [default backend](#) for users to have a quick start, unless it is being overwritten by ingress rules.

Customized configuration

As an alternative to the basic configuration presented in the above section, the next set of steps will show how to deploy a customized ingress controller. You'll have the option of using an internal static IP address, or using a dynamic public IP address.

Import the images used by the Helm chart into your ACR

Azure CLI

To control image versions, you'll want to import them into your own Azure Container Registry. The [NGINX ingress controller Helm chart](#) relies on three container images. Use `az acr import` to import those images into your ACR.

Azure CLI

```
REGISTRY_NAME=<REGISTRY_NAME>
SOURCE_REGISTRY=registry.k8s.io
CONTROLLER_IMAGE=ingress-nginx/controller
CONTROLLER_TAG=v1.8.1
PATCH_IMAGE=ingress-nginx/kube-webhook-certgen
PATCH_TAG=v20230407
DEFAULTBACKEND_IMAGE=defaultbackend-amd64
DEFAULTBACKEND_TAG=1.5

az acr import --name $REGISTRY_NAME --source
$SOURCE_REGISTRY/$CONTROLLER_IMAGE:$CONTROLLER_TAG --image
$CONTROLLER_IMAGE:$CONTROLLER_TAG
az acr import --name $REGISTRY_NAME --source
$SOURCE_REGISTRY/$PATCH_IMAGE:$PATCH_TAG --image $PATCH_IMAGE:$PATCH_TAG
az acr import --name $REGISTRY_NAME --source
```

```
$SOURCE_REGISTRY/$DEFAULTBACKEND_IMAGE:$DEFAULTBACKEND_TAG --image  
$DEFAULTBACKEND_IMAGE:$DEFAULTBACKEND_TAG
```

ⓘ Note

In addition to importing container images into your ACR, you can also import Helm charts into your ACR. For more information, see [Push and pull Helm charts to an Azure Container Registry](#).

Create an ingress controller

To create the ingress controller, use Helm to install *ingress-nginx*. The ingress controller needs to be scheduled on a Linux node. Windows Server nodes shouldn't run the ingress controller. A node selector is specified using the `--set nodeSelector` parameter to tell the Kubernetes scheduler to run the NGINX ingress controller on a Linux-based node.

For added redundancy, two replicas of the NGINX ingress controllers are deployed with the `--set controller.replicaCount` parameter. To fully benefit from running replicas of the ingress controller, make sure there's more than one node in your AKS cluster.

The following example creates a Kubernetes namespace for the ingress resources named *ingress-basic* and is intended to work within that namespace. Specify a namespace for your own environment as needed. If your AKS cluster isn't Kubernetes role-based access control enabled, add `--set rbac.create=false` to the Helm commands.

ⓘ Note

If you would like to enable [client source IP preservation](#) for requests to containers in your cluster, add `-set controller.service.externalTrafficPolicy=Local` to the Helm install command. The client source IP is stored in the request header under *X-Forwarded-For*. When you're using an ingress controller with client source IP preservation enabled, TLS pass-through won't work.

Azure CLI

Console

```

# Add the ingress-nginx repository
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
helm repo update

# Set variable for ACR location to use for pulling images
ACR_LOGIN_SERVER=<REGISTRY_LOGIN_SERVER>

# Use Helm to deploy an NGINX ingress controller
helm install ingress-nginx ingress-nginx/ingress-nginx \
--version 4.7.1 \
--namespace ingress-basic \
--create-namespace \
--set controller.replicaCount=2 \
--set controller.nodeSelector."kubernetes\.io/os"=linux \
--set controller.image.registry=$ACR_LOGIN_SERVER \
--set controller.image.image=$CONTROLLER_IMAGE \
--set controller.image.tag=$CONTROLLER_TAG \
--set controller.image.digest="" \
--set
controller.admissionWebhooks.patch.nodeSelector."kubernetes\.io/os"=linu
x \
--set
controller.service.annotations."service\.beta\.kubernetes\.io/azure-
load-balancer-health-probe-request-path"/healthz \
--set controller.service.externalTrafficPolicy=Local \
--set
controller.admissionWebhooks.patch.image.registry=$ACR_LOGIN_SERVER \
--set controller.admissionWebhooks.patch.image.image=$PATCH_IMAGE \
--set controller.admissionWebhooks.patch.image.tag=$PATCH_TAG \
--set controller.admissionWebhooks.patch.image.digest="" \
--set defaultBackend.nodeSelector."kubernetes\.io/os"=linux \
--set defaultBackend.image.registry=$ACR_LOGIN_SERVER \
--set defaultBackend.image.image=$DEFAULTBACKEND_IMAGE \
--set defaultBackend.image.tag=$DEFAULTBACKEND_TAG \
--set defaultBackend.image.digest=""

```

Create an ingress controller using an internal IP address

By default, an NGINX ingress controller is created with a dynamic public IP address assignment. A common configuration requirement is to use an internal, private network and IP address. This approach allows you to restrict access to your services to internal users, with no external access.

Use the `--set controller.service.loadBalancerIP` and `--set`

`controller.service.annotations."service\.beta\.kubernetes\.io/azure-load-balancer-
internal"=true` parameters to assign an internal IP address to your ingress controller.

Provide your own internal IP address for use with the ingress controller. Make sure that this IP address isn't already in use within your virtual network. If you're using an existing

virtual network and subnet, you must configure your AKS cluster with the correct permissions to manage the virtual network and subnet. For more information, see [Use kubenet networking with your own IP address ranges in Azure Kubernetes Service \(AKS\)](#) or [Configure Azure CNI networking in Azure Kubernetes Service \(AKS\)](#).

Azure CLI

Console

```
# Add the ingress-nginx repository
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
helm repo update

# Set variable for ACR location to use for pulling images
ACR_LOGIN_SERVER=<REGISTRY_LOGIN_SERVER>

# Use Helm to deploy an NGINX ingress controller
helm install ingress-nginx ingress-nginx/ingress-nginx \
--version 4.7.1 \
--namespace ingress-basic \
--create-namespace \
--set controller.replicaCount=2 \
--set controller.nodeSelector."kubernetes\\.io/os"=linux \
--set controller.image.registry=$ACR_LOGIN_SERVER \
--set controller.image.image=$CONTROLLER_IMAGE \
--set controller.image.tag=$CONTROLLER_TAG \
--set controller.image.digest="" \
--set
controller.admissionWebhooks.patch.nodeSelector."kubernetes\\.io/os"=linu
x \
--set controller.service.loadBalancerIP=10.224.0.42 \
--set
controller.service.annotations."service\\.beta\\.kubernetes\\.io/azure-
load-balancer-internal"=true \
--set
controller.service.annotations."service\\.beta\\.kubernetes\\.io/azure-
load-balancer-health-probe-request-path"=/healthz \
--set
controller.admissionWebhooks.patch.image.registry=$ACR_LOGIN_SERVER \
--set controller.admissionWebhooks.patch.image.image=$PATCH_IMAGE \
--set controller.admissionWebhooks.patch.image.tag=$PATCH_TAG \
--set controller.admissionWebhooks.patch.image.digest="" \
--set defaultBackend.nodeSelector."kubernetes\\.io/os"=linux \
--set defaultBackend.image.registry=$ACR_LOGIN_SERVER \
--set defaultBackend.image.image=$DEFAULTBACKEND_IMAGE \
--set defaultBackend.image.tag=$DEFAULTBACKEND_TAG \
--set defaultBackend.image.digest=""
```

Check the load balancer service

Check the load balancer service by using `kubectl get services`.

Console

```
kubectl get services --namespace ingress-basic -o wide -w ingress-nginx-controller
```

When the Kubernetes load balancer service is created for the NGINX ingress controller, an IP address is assigned under *EXTERNAL-IP*, as shown in the following example output:

Console

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP
PORT(S)	AGE	SELECTOR	
ingress-nginx-controller	LoadBalancer	10.0.65.205	EXTERNAL-IP
80:30957/TCP,443:32414/TCP	1m		
		app.kubernetes.io/component=controller,app.kubernetes.io/instance=ingress-nginx,app.kubernetes.io/name=ingress-nginx	

If you browse to the external IP address at this stage, you see a 404 page displayed. This is because you still need to set up the connection to the external IP, which is done in the next sections.

Run demo applications

To see the ingress controller in action, run two demo applications in your AKS cluster. In this example, you use `kubectl apply` to deploy two instances of a simple *Hello world* application.

1. Create an `aks-helloworld-one.yaml` file and copy in the following example YAML:

YAML

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: aks-helloworld-one
spec:
  replicas: 1
  selector:
    matchLabels:
      app: aks-helloworld-one
  template:
    metadata:
      labels:
```

```

        app: aks-helloworld-one
spec:
  containers:
    - name: aks-helloworld-one
      image: mcr.microsoft.com/azuredocs/aks-helloworld:v1
      ports:
        - containerPort: 80
      env:
        - name: TITLE
          value: "Welcome to Azure Kubernetes Service (AKS)"
---
apiVersion: v1
kind: Service
metadata:
  name: aks-helloworld-one
spec:
  type: ClusterIP
  ports:
    - port: 80
  selector:
    app: aks-helloworld-one

```

2. Create an `aks-helloworld-two.yaml` file and copy in the following example YAML:

YAML

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: aks-helloworld-two
spec:
  replicas: 1
  selector:
    matchLabels:
      app: aks-helloworld-two
  template:
    metadata:
      labels:
        app: aks-helloworld-two
    spec:
      containers:
        - name: aks-helloworld-two
          image: mcr.microsoft.com/azuredocs/aks-helloworld:v1
          ports:
            - containerPort: 80
          env:
            - name: TITLE
              value: "AKS Ingress Demo"
---
apiVersion: v1
kind: Service
metadata:
  name: aks-helloworld-two

```

```
spec:  
  type: ClusterIP  
  ports:  
    - port: 80  
  selector:  
    app: aks-helloworld-two
```

- Run the two demo applications using `kubectl apply`:

Console

```
kubectl apply -f aks-helloworld-one.yaml --namespace ingress-basic  
kubectl apply -f aks-helloworld-two.yaml --namespace ingress-basic
```

Create an ingress route

Both applications are now running on your Kubernetes cluster. To route traffic to each application, create a Kubernetes ingress resource. The ingress resource configures the rules that route traffic to one of the two applications.

In the following example, traffic to `EXTERNAL_IP/hello-world-one` is routed to the service named `aks-helloworld-one`. Traffic to `EXTERNAL_IP/hello-world-two` is routed to the `aks-helloworld-two` service. Traffic to `EXTERNAL_IP/static` is routed to the service named `aks-helloworld-one` for static assets.

- Create a file named `hello-world-ingress.yaml` and copy in the following example YAML:

YAML

```
apiVersion: networking.k8s.io/v1  
kind: Ingress  
metadata:  
  name: hello-world-ingress  
  annotations:  
    nginx.ingress.kubernetes.io/ssl-redirect: "false"  
    nginx.ingress.kubernetes.io/use-regex: "true"  
    nginx.ingress.kubernetes.io/rewrite-target: /$2  
spec:  
  ingressClassName: nginx  
  rules:  
    - http:  
        paths:  
          - path: /hello-world-one(/|$(.))*  
            pathType: Prefix  
            backend:  
              service:
```

```

        name: aks-helloworld-one
        port:
          number: 80
      - path: /hello-world-two(/|$(.))
        pathType: Prefix
        backend:
          service:
            name: aks-helloworld-two
            port:
              number: 80
      - path: /(.*)
        pathType: Prefix
        backend:
          service:
            name: aks-helloworld-one
            port:
              number: 80
    ---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: hello-world-ingress-static
  annotations:
    nginx.ingress.kubernetes.io/ssl-redirect: "false"
    nginx.ingress.kubernetes.io/rewrite-target: /static/$2
spec:
  ingressClassName: nginx
  rules:
  - http:
      paths:
      - path: /static(/|$(.))
        pathType: Prefix
        backend:
          service:
            name: aks-helloworld-one
            port:
              number: 80

```

2. Create the ingress resource using the `kubectl apply` command.

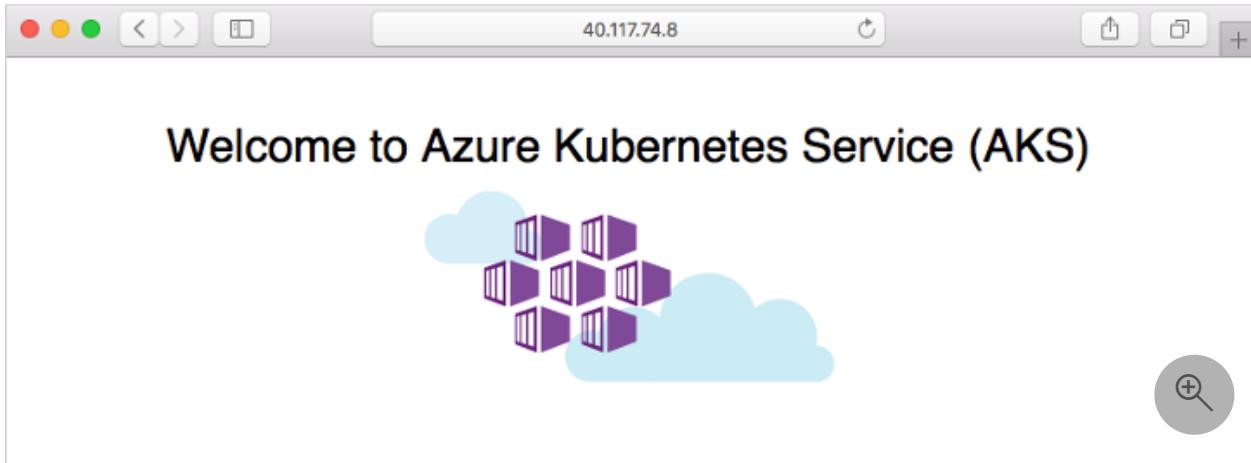
Console

```
kubectl apply -f hello-world-ingress.yaml --namespace ingress-basic
```

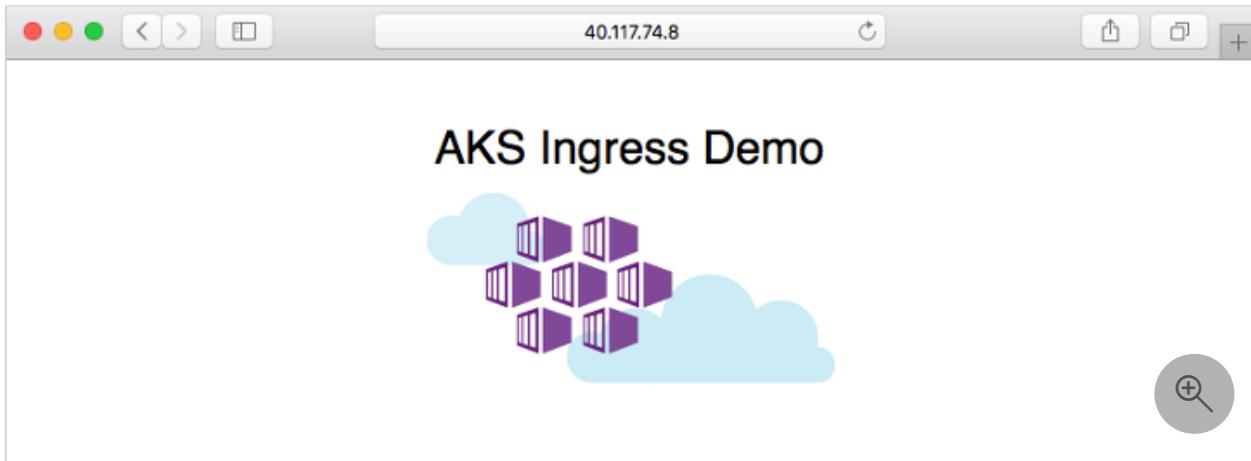
Test the ingress controller

To test the routes for the ingress controller, browse to the two applications. Open a web browser to the IP address of your NGINX ingress controller, such as *EXTERNAL_IP*. The

first demo application is displayed in the web browser, as shown in the following example:



Now add the `/hello-world-two` path to the IP address, such as `EXTERNAL_IP/hello-world-two`. The second demo application with the custom title is displayed:



Test an internal IP address

1. Create a test pod and attach a terminal session to it.

```
Console

kubectl run -it --rm aks-ingress-test --
image=mcr.microsoft.com/dotnet/runtime-deps:6.0 --namespace ingress-
basic
```

2. Install `curl` in the pod using `apt-get`.

```
Console

apt-get update && apt-get install -y curl
```

3. Access the address of your Kubernetes ingress controller using `curl`, such as <http://10.224.0.42>. Provide your own internal IP address specified when you deployed the ingress controller.

```
Console
```

```
curl -L http://10.224.0.42
```

No path was provided with the address, so the ingress controller defaults to the `/` route. The first demo application is returned, as shown in the following condensed example output:

```
Console
```

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <link rel="stylesheet" type="text/css" href="/static/default.css">
    <title>Welcome to Azure Kubernetes Service (AKS)</title>
[...]
```

4. Add the `/hello-world-two` path to the address, such as <http://10.224.0.42/hello-world-two>.

```
Console
```

```
curl -L -k http://10.224.0.42/hello-world-two
```

The second demo application with the custom title is returned, as shown in the following condensed example output:

```
Console
```

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <link rel="stylesheet" type="text/css" href="/static/default.css">
    <title>AKS Ingress Demo</title>
[...]
```

Clean up resources

This article used Helm to install the ingress components and sample apps. When you deploy a Helm chart, many Kubernetes resources are created. These resources include pods, deployments, and services. To clean up these resources, you can either delete the entire sample namespace, or the individual resources.

Delete the sample namespace and all resources

To delete the entire sample namespace, use the `kubectl delete` command and specify your namespace name. All the resources in the namespace are deleted.

```
Console
```

```
kubectl delete namespace ingress-basic
```

Delete resources individually

Alternatively, a more granular approach is to delete the individual resources created.

1. List the Helm releases with the `helm list` command.

```
Console
```

```
helm list --namespace ingress-basic
```

Look for charts named *ingress-nginx* and *aks-helloworld*, as shown in the following example output:

```
Console
```

NAME	CHART	NAMESPACE	REVISION	UPDATED
STATUS			APP VERSION	
ingress-nginx		ingress-basic	1	2020-01-06
19:55:46.358275 -0600 CST		deployed		nginx-ingress-1.27.1
0.26.1				

2. Uninstall the releases with the `helm uninstall` command.

```
Console
```

```
helm uninstall ingress-nginx --namespace ingress-basic
```

3. Remove the two sample applications.

Console

```
kubectl delete -f aks-helloworld-one.yaml --namespace ingress-basic  
kubectl delete -f aks-helloworld-two.yaml --namespace ingress-basic
```

4. Remove the ingress route that directed traffic to the sample apps.

Console

```
kubectl delete -f hello-world-ingress.yaml
```

5. Delete the namespace using the `kubectl delete` command and specifying your namespace name.

Console

```
kubectl delete namespace ingress-basic
```

Next steps

To configure TLS with your existing ingress components, see [Use TLS with an ingress controller](#).

To configure your AKS cluster to use application routing, see [Application routing add-on](#).

This article included some external components to AKS. To learn more about these components, see the following project pages:

- [Helm CLI](#)
- [NGINX ingress controller](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Provide product feedback ↗

Troubleshoot Application Gateway Ingress Controller connectivity issues

07/05/2025

The [Application Gateway Ingress Controller \(AGIC\)](#) is a Kubernetes application that enables Azure Kubernetes Service (AKS) users to use Azure's native Application Gateway L7 load-balancer to expose cloud software to the internet.

This article provides step-by-step guidance to troubleshoot AGIC connectivity issues effectively.

Prerequisites

Before you start, make sure that you have the following tools installed:

- **Azure CLI:** Follow the [installation guide](#).
- **Kubernetes CLI (`kubectl`):** Use Azure CLI to install it by running the command, `az aks install-cli`.
- **Client URL (`cURL`) tool:** Install it by following [this guidance](#).

Common symptoms

Note

This article focuses on Application Gateway Ingress Controller issues. Other underlying problems might cause similar symptoms. For more information, see [Troubleshoot connection issues to an app hosted in an AKS cluster](#).

 Expand table

Symptom	Description
Ingress without IP address	Errors in assigning an <code>IP address</code> to the <code>Ingress</code> indicate that AGIC isn't functioning correctly.
HTTP Timeout	If <code>DNS</code> , <code>Ingress</code> , and <code>Application</code> are working, AGIC is the likely cause of the issue.

Step 1: Verify application functionality

Make sure that your application is functioning correctly before you troubleshoot AGIC. Follow these steps:

1. Describe your service:

```
Console
```

```
kubectl describe service <YOUR_SERVICE> -n <YOUR_NAMESPACE>
```

2. Copy the port details:

```
Console
```

```
$ kubectl describe service <YOUR_SERVICE> -n <YOUR_NAMESPACE>
Name:           dummy-web
Namespace:      default
Labels:         app=dummy-web
Annotations:    <none>
Selector:       app=dummy-web
Type:          ClusterIP
IP Family Policy: SingleStack
IP Families:   IPv4
IP:            10.0.29.113
IPs:           10.0.29.113
Port:          <unset>  8080/TCP
TargetPort:     8080/TCP
Endpoints:     10.224.0.49:8080,10.224.0.47:8080,10.224.0.4:8080 +
               12 more...
Session Affinity: None
Internal Traffic Policy: Cluster
Events:
Type  Reason          Age   From
Message
----  -----          ----  -----
---
```

3. Port-forward your service:

```
Console
```

```
kubectl port-forward svc/<YOUR_SERVICE> 9090:<YOUR_SERVICE_PORT> -n
<YOUR_NAMESPACE>
```

4. Test the application locally:

```
Console
```

```
curl -v http://localhost:9090
```

5. Verify application functionality:

ⓘ Note

Investigate and resolve any errors that you encountered during this step before you proceed.

Console

```
$ curl -v http://localhost:9090
* Host localhost:9090 was resolved.
* IPv6: ::1
* IPv4: 127.0.0.1
* Trying [::1]:9090...
* Connected to localhost (::1) port 9090
> GET / HTTP/1.1
> Host: localhost:9090
> User-Agent: curl/8.5.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: text/html; charset=utf-8
< Date: Tue, 27 May 2025 00:54:58 GMT
< Server: Kestrel
< Transfer-Encoding: chunked
```

Step 2: Inspect Ingress settings

Verify that the `Ingress` was created correctly:

1. Describe the specific Ingress:

Console

```
kubectl describe ingress <YOUR_INGRESS> -n <YOUR_NAMESPACE>
```

2. Check events, rules, and address:

Console

```
$ kubectl describe ingress <YOUR_INGRESS> -n <YOUR_NAMESPACE>
Name:           dummy-web
Labels:         <none>
Namespace:     default
Address:
Ingress Class: azure-application-gateway
```

```
Default backend: <default>
Rules:
Host      Path  Backends
----      ----  -----
*
      /    dummy-web:8080
(10.224.0.70:8080,10.224.0.72:8080,10.224.0.88:8080 + 12 more...)
Annotations: <none>
Events:
Type    Reason          Age           From
Message
----    -----          ----
-----
Normal  ResetIngressStatus  13m (x5 over 13m)  azure/application-gateway
Reset IP for Ingress default/dummy-web. Application Gateway
<APPLICATION_GATEWAY_ID> is in stopped state
```

If the `Ingress` lacks an address or displays events that indicate issues, investigate further.

Step 3: Inspect Ingress pod logs

1. Find the Ingress pod:

```
Console

kubectl get pod -A | grep ingress
```

2. Inspect the logs:

```
Console

kubectl logs <INGRESS_POD_NAME> -n <YOUR_NAMESPACE>
```

For the AGIC that is deployed by using the add-on, run the following command:

```
Console

kubectl logs -n kube-system -l=app=ingress-appgw
```

Look for any errors or warnings that might indicate what's going wrong.

Step 4: Check Application Gateway operational State

It focuses on understanding the operational state of the [Application Gateway](#) if it's used as an [Ingress Controller on AKS](#).

Add-on

1. Get the Application Gateway name:

Console

```
az aks show --name <YOUR_AKS_NAME> --resource-group <YOUR_RG_NAME> --query addonProfiles.ingressApplicationGateway
```

① Note

If you encounter an unexpected error during this step, AGIC might be misconfigured. In this case, refer to the following guide: [Enable the ingress controller add-on for a new AKS cluster with a new application gateway instance](#).

Console

```
{
  "config": {
    "applicationGatewayName": "<YOUR_APPLICATION_GATEWAY_NAME>",
    "effectiveApplicationGatewayId": "...",
    "subnetCIDR": "..."
  },
  "enabled": true,
  ..
}
```

2. Verify the Application Gateway operational state:

Console

```
az network application-gateway show --name
<YOUR_APPLICATION_GATEWAY_NAME> --resource-group <YOUR_RG_NAME> --query
operationalState
```

Step 5 (Optional): Inspect Mapped Kubernetes and Application Gateway IPs

The AGIC monitors the pod IPs and maps them to `backendAddressPools` in the `Application Gateway` instance. This step verifies that integration.

1. Get the Application Gateway `backendAddressPools`:

Console

```
az network application-gateway show --name <YOUR_APPLICATION_GATEWAY_NAME> --resource-group <YOUR_RG_NAME> --query backendAddressPools
```

2. Get the pod IPs by using Kubernetes endpoints:

Console

```
kubectl describe endpoints <YOUR_SERVICE_NAME> -n <YOUR_NAMESPACE> | grep Addresses
```

3. Compare the results:

Make sure that the lists from steps 1 and 2 are equivalent. If they're not, AGIC might not be working correctly.

Solution: Start the Application Gateway

If AGIC isn't working as expected, it might be stopped or misconfigured. If the Application Gateway operational state isn't `Running`, start or restart AGIC, wait a few seconds, and then test the application again.

Console

```
az network application-gateway start --name <YOUR_APPLICATION_GATEWAY_NAME> --resource-group <YOUR_RG_NAME>
```

Additional resources

- [Learn more about Azure Kubernetes Service \(AKS\) best practices](#)
- [Monitor your Kubernetes cluster performance with Container insights](#)

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Troubleshoot API server and etcd problems in Azure Kubernetes Services

07/23/2025

This guide is designed to help you identify and resolve any unlikely problems that you might encounter within the API server in large Microsoft Azure Kubernetes Services (AKS) deployments.

Microsoft has tested the reliability and performance of the API server at a scale of 5,000 nodes and 200,000 pods. The cluster that contains the API server has the ability to automatically scale out and deliver [Kubernetes Service Level Objectives \(SLOs\)](#). If you experience high latencies or time-outs, it's probably because there's a resource leakage on the distributed `etc` directory (etcd), or an offending client has excessive API calls.

Prerequisites

- The [Azure CLI](#).
- The Kubernetes [kubectl](#) tool. To install kubectl by using Azure CLI, run the `az aks install-cli` command.
- AKS diagnostics logs (specifically, kube-audit events) that are enabled and sent to a [Log Analytics workspace](#). To determine if logs are collected using [resource-specific](#) or [Azure diagnostics](#) mode, check the [Diagnostic Settings](#) blade in the Azure portal.
- The Standard tier for AKS clusters. If you're using the Free tier, the API server and etcd contain limited resources. AKS clusters in the Free tier don't provide high availability. This is often the root cause of API server and etcd problems.
- The [kubectl-aks](#) plugin for running commands directly on AKS nodes without using the Kubernetes control plane.

Basic health checks

- Resource health events

AKS provides Resource health events for critical component downtime. Before proceeding, ensure there are no critical events reported in [Resource Health](#).

Date	Description
✓ 08/03/2023	⚠ 1 health event(s)
	⚠ Degraded (Unplanned) : Etcd is Running out of Storage Space (Unplanned)
	At Thursday, August 3, 2023 at 12:42:26 AM PDT, the Azure monitoring system received the following information regarding your Azure Kubernetes Service (AKS):
	Your ETCD database usage is high. To reduce event storage usage, use the recommended actions listed below.
00:42:26 (PDT) - Ongoing	<p>Recommended Steps</p> <ul style="list-style-type: none"> • Delete objects that are no longer required, such as completed jobs, old deployment versions, etc. • For objects that support automatic clean-up, like Jobs, you can set Time to live (TTL) values to limit the lifetime of these objects. • You can also automatically enforce limits on the number of objects by defining object quotas. • Please check the ETCD Troubleshooting documentation for further details.
	Download as PDF



- Diagnose and solve problems

AKS provides a dedicated troubleshooting category for Cluster and Control Plane Availability and Performance.

The screenshot shows the Azure portal's 'Diagnose and solve problems' interface. On the left, there's a sidebar with various navigation options like Overview, Activity log, Tags, and Kubernetes resources. The 'Diagnose and solve problems' option is selected. In the main area, there's a 'Risk alerts' section showing 1 Warning and 4 Success. Below it is a 'Troubleshooting categories' section. The 'Cluster and Control Plane Availability and Performance' category is highlighted with a red box. Other categories shown include Connectivity Issues, Create, Upgrade, Delete and Scale, Identity and Security, and Best Practices. Each category has a brief description and a 'Troubleshoot' button.



Symptoms

The following table outlines the common symptoms of API server failures:

[Expand table](#)

Symptom	Description
Time-outs from the API server	Frequent time-outs that are beyond the guarantees in the AKS API server SLA . For example, <code>kubectl</code> commands time-out.
High latencies	High latencies that make the Kubernetes SLOs fail. For example, the <code>kubectl</code> command takes more than 30 seconds to list pods.
API server pod in <code>CrashLoopbackOff</code> status or facing webhook call failures	Verify that you don't have any custom admission webhook (such as the Kyverno policy engine) that's blocking the calls to the API server.

Troubleshooting checklist

If you are experiencing high latency times, follow these steps to pinpoint the offending client and the types of API calls that fail.

Step 1: Identify top user agents by the number of requests

To identify which clients generate the most requests (and potentially the most API server load), run a query that resembles the following code. The following query lists the top 10 user agents by the number of API server requests sent.

Resource-specific

Kusto

```
AKSAudit
| where TimeGenerated between(now(-1h)..now()) // When you experienced the
problem
| summarize count() by UserAgent
| top 10 by count_
| project UserAgent, count_
```

(!) Note

If your query returns no results, you may have selected the wrong table to query diagnostics logs. In resource-specific mode, data is written to individual tables depending on the category of the resource. Diagnostics logs are written to the `AKSAudit` table. In Azure diagnostics mode, all data is written to the `AzureDiagnostics` table. For more information, see [Azure resource logs](#).

Although it's helpful to know which clients generate the highest request volume, high request volume alone might not be a cause for concern. A better indicator of the actual load that each client generates on the API server is the response latency that they experience.

Step 2: Identify and chart the average latency of API server requests per user agent

- 1.a. Use the API Server Resource Intensive Listing Detector in Azure Portal

New: Azure Kubernetes Service now provides a built-in analyzer to help you identify agents making resource-intensive LIST calls, which are a leading cause of API server and etcd performance issues.

How to access the detector:

1. Open your AKS cluster in the Azure portal.
2. Go to **Diagnose and solve problems**.
3. Click **Cluster and Control Plane Availability and Performance**.
4. Select **API server resource intensive listing detector**.

This detector analyzes recent API server activity and highlights agents or workloads generating large or frequent LIST calls. It provides a summary of potential impacts, such as request timeouts, increased 408/503 errors, node instability, health probe failures, and OOM-Kills in API server or etcd.

How to interpret the detector output

- **Summary:**

Indicates if resource-intensive LIST calls were detected and describes possible impacts on your cluster.

- **Analysis window:**

Shows the 30-minute window analyzed, with peak memory and CPU usage.

- **Read types:**

Explains whether LIST calls were served from the API server cache (preferred) or required fetching from etcd (most impactful).

- **Charts and tables:**

Identify which agents, namespaces, or workloads are generating the most resource-intensive LIST calls.

Only successful LIST calls are counted. Failed or throttled calls are excluded.

The analyzer also provides actionable recommendations directly in the Azure portal, tailored to the detected patterns, to help you remediate and optimize your cluster.

Note

The API server resource intensive listing detector is available to all users with access to the AKS resource in the Azure portal. No special permissions or prerequisites are required.

After identifying the offending agents and applying the above recommendations, you can further use [Priority and Fairness](#) or refer to [this section](#) to throttle or isolate problematic clients.

1.b. Additionally, you can also run following query to identify the average latency of API server requests per user agent as plotted on a time chart:

Resource-specific

Kusto

```
AKSAudit
| where TimeGenerated between(now(-1h)..now()) // When you experienced the
  problem
| extend start_time = RequestReceivedTime
| extend end_time = StageReceivedTime
| extend latency = datetime_diff('millisecond', end_time, start_time)
| summarize avg(latency) by UserAgent, bin(start_time, 5m)
| render timechart
```

This query is a follow-up to the query in the "[Identify top user agents by the number of requests](#)" section. It might give you more insights into the actual load that's generated by each user agent over time.

💡 Tip

By analyzing this data, you can identify patterns and anomalies that can indicate problems on your AKS cluster or applications. For example, you might notice that a particular user is experiencing high latency. This scenario can indicate the type of API calls that are causing excessive load on the API server or etcd.

Step 3: Identify bad API calls for a given user agent

Run the following query to tabulate the 99th percentile (P99) latency of API calls across different resource types for a given client:

Resource-specific

Kusto

```
AKSAudit
| where TimeGenerated between(now(-1h)..now()) // When you experienced the
```

```
problem
| extend HttpMethod = Verb
| extend Resource = tostring(ObjectRef.resource)
| where UserAgent == "DUMMYUSERAGENT" // Filter by name of the useragent you
are interested in
| where Resource != ""
| extend start_time = RequestReceivedTime
| extend end_time = StageReceivedTime
| extend latency = datetime_diff('millisecond', end_time, start_time)
| summarize p99latency=percentile(latency, 99) by HttpMethod, Resource
| render table
```

The results from this query can be useful to identify the kinds of API calls that fail the upstream Kubernetes SLOs. In most cases, an offending client might be making too many `LIST` calls on a large set of objects or objects that are too large. Unfortunately, no hard scalability limits are available to guide users about API server scalability. API server or etcd scalability limits depend on various factors that are explained in [Kubernetes Scalability thresholds](#).

Cause 1: A network rule blocks the traffic from agent nodes to the API server

A network rule can block traffic between the agent nodes and the API server.

To verify whether a misconfigured network policy is blocking communication between the API server and agent nodes, run the following `kubectl-aks` commands:

Bash

```
kubectl aks config import \
--subscription <mySubscriptionID> \
--resource-group <myResourceGroup> \
--cluster-name <myAKSCluster>

kubectl aks check-apiserver-connectivity --node <myNode>
```

The `config import` command retrieves the Virtual Machine Scale Set information for all the nodes in the cluster. Then, the `check-apiserver-connectivity` command uses this information to verify the network connectivity between the API server and a specified node, specifically for its underlying scale set instance.

(!) Note

If the output of the `check-apiserver-connectivity` command contains the `Connectivity check: succeeded` message, then the network connectivity is unimpeded.

Solution 1: Fix the network policy to remove the traffic blockage

If the command output indicates that a connection failure occurred, reconfigure the network policy so that it doesn't unnecessarily block traffic between the agent nodes and the API server.

Cause 2: An offending client leaks etcd objects and results in a slowdown of etcd

A common problem is continuously creating objects without deleting unused ones in the etcd database. This can cause performance problems when etcd deals with too many objects (more than 10,000) of any type. A rapid increase of changes on such objects could also cause the etcd database size (4 gigabytes by default) to be exceeded.

To check the etcd database usage, navigate to **Diagnose and Solve problems** in the Azure portal. Run the **Etcd Availability Issues** diagnosis tool by searching for "etcd" in the search box. The diagnosis tool shows you the usage breakdown and the total database size.

The screenshot shows the Azure Kubernetes Service Diagnostics (Preview) interface. On the left, there's a sidebar with navigation links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems (which is selected), Microsoft Defender for Cloud, Kubernetes resources, Settings, Monitoring, Automation, and Help. The main area has a search bar and a feedback link. Below the search bar, there's a heading "Azure Kubernetes Service Diagnostics (Preview) - Investigate how your cluster is performing, diagnose issues, and discover how to improve its reliability." A search bar shows the term "etcd". A dropdown menu for "etcd" lists "Etcd Availability Issues" under "Known Issues, Availability and Performance". Below this, there are four troubleshooting sections: "Performance" (with a "Troubleshoot" button), "Networking Issues" (with a "Troubleshoot" button), "Create, Upgrade, Delete and Scale" (with a "Troubleshoot" button), and "Storage" (with a "Troubleshoot" button). At the bottom right is a magnifying glass icon with a plus sign.

If you just want a quick way to view the current size of your etcd database in bytes, run the following command:

Bash

```
kubectl get --raw /metrics | grep -E  
"etcd_db_total_size_in_bytes|apiserver_storage_size_bytes|apiserver_storage_db_tot  
al_size_in_bytes"
```

! Note

The metric name in the previous command is different for different Kubernetes versions.

For Kubernetes 1.25 and earlier, use `etcd_db_total_size_in_bytes`. For Kubernetes 1.26 to 1.28, use `apiserver_storage_db_total_size_in_bytes`.

Solution 2: Define quotas for object creation, delete objects, or limit object lifetime in etcd

To prevent etcd from reaching capacity and causing cluster downtime, you can limit the maximum number of resources that are created. You can also slow the number of revisions that are generated for resource instances. To limit the number of objects that can be created, you can [define object quotas](#).

If you have identified objects that are no longer in use but are taking up resources, consider deleting them. For example, you can delete completed jobs to free up space:

Bash

```
kubectl delete jobs --field-selector status.successful=1
```

For objects that support [automatic cleanup](#), you can set Time to Live (TTL) values to limit the lifetime of these objects. You can also label your objects so that you can bulk delete all the objects of a specific type by using label selectors. If you establish [owner references](#) among objects, any dependent objects are automatically deleted after the parent object is deleted.

Cause 3: An offending client makes excessive LIST or PUT calls

If you determine that etcd isn't overloaded with too many objects, an offending client might be making too many `LIST` or `PUT` calls to the API server.

Solution 3a: Tune your API call pattern

Consider tuning your client's API call pattern to reduce the pressure on the control plane.

Solution 3b: Throttle a client that's overwhelming the control plane

If you can't tune the client, you can use the [Priority and Fairness](#) feature in Kubernetes to throttle the client. This feature can help preserve the health of the control plane and prevent other applications from failing.

The following procedure shows you how to throttle an offending client's LIST Pods API set to five concurrent calls:

1. Create a [FlowSchema](#) that matches the API call pattern of the offending client:

YAML

```
apiVersion: flowcontrol.apiserver.k8s.io/v1beta2
kind: FlowSchema
metadata:
  name: restrict-bad-client
spec:
  priorityLevelConfiguration:
    name: very-low-priority
  distinguisherMethod:
    type: ByUser
  rules:
    - resourceRules:
        - apiGroups: []
          namespaces: ["default"]
          resources: ["pods"]
          verbs: ["list"]
    subjects:
      - kind: ServiceAccount
        serviceAccount:
          name: bad-client-account
          namespace: default
```

2. Create a lower priority configuration to throttle bad API calls of the client:

YAML

```
apiVersion: flowcontrol.apiserver.k8s.io/v1beta2
kind: PriorityLevelConfiguration
metadata:
  name: very-low-priority
spec:
  limited:
    assuredConcurrencyShares: 5
    limitResponse:
```

```
type: Reject  
type: Limited
```

3. Observe the throttled call in the API server metrics.

```
Bash
```

```
kubectl get --raw /metrics | grep "restrict-bad-client"
```

Cause 4: A custom webhook might cause a deadlock in API server pods

A custom webhook, such as Kyverno, might be causing a deadlock within API server pods.

Check the events that are related to your API server. You might see event messages that resemble the following text:

```
| Internal error occurred: failed calling webhook "mutate.kyverno.svc-fail": failed to call  
| webhook: Post "https://kyverno-system-kyverno-system-svc.kyverno-  
| system.svc:443/mutate/fail?timeout=10s": write unix @->/tunnel-uds/proxysocket: write:  
| broken pipe
```

In this example, the validating webhook is blocking the creation of some API server objects. Because this scenario might occur during bootstrap time, the API server and Konnectivity pods can't be created. Therefore, the webhook can't connect to those pods. This sequence of events causes the deadlock and the error message.

Solution 4: Delete webhook configurations

To fix this problem, delete the validating and mutating webhook configurations. To delete these webhook configurations in Kyverno, review the [Kyverno troubleshooting article](#).

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the

performance or reliability of these products.

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Improve container image pull performance in Azure Kubernetes Service

Article • 10/25/2024

This article provides methods to improve container image pull performance in Azure Kubernetes Service (AKS).

Serialized and parallel image pulls

There are two types of container image pulls: serialized and parallel image pulls.

In AKS versions earlier than 1.31, by default, AKS enables serialized image pulls. Starting in AKS version 1.31 preview, by default, AKS pulls container images in parallel. Typically, serialized image pulls should be less performant than parallel image pulls, particularly when the service tries to pull large or numerous container images.

However, you might still experience decreased performance by using parallel image pulls compared to using serial image pulls. Trying to pull large or numerous images in a parallel manner might throttle the disk. This is especially true when you use unoptimized disk and VM resources. In this situation, you might notice that the time to pull the first images is faster in a serial setup, but the time to pull all images is faster in a parallel setup. Depending on your workload needs, you should consider using the following steps to scale your disk and VM resources or toggle the image pull type.

Improve image pull performance in AKS versions earlier than 1.31

Upgrade to AKS 1.31 to switch from serialized image pulls to parallel image pulls. Parallel image pulls generally improve image pull performance.

Improve image pull performance in AKS 1.31 and later versions

If you notice that parallel image pulls increase the latency of operations compared to serialized image pulls, try one or more methods below to improve performance:

- Use ephemeral OS disks
- Increase the size of the OS disk
- Use newer VM SKUs
- Increase the size of the VM SKU
- Toggle image pull type

Use ephemeral OS disks

Change from managed operating system (OS) disks to ephemeral OS disks to reduce disk throughput bottlenecks in image pulls. Ephemeral OS disks provide a dynamic amount of available IOPS, allowing them to scale based on your workload requirements.

Increase the size of the OS disk

Increase the size of your OS disk to reduce a disk throughput bottleneck in image pulling. Larger managed OS disks increase the available disk bandwidth.

Use newer VM SKUs

Switch to newer hardware generation virtual machine (VM) stock keeping units (SKUs), such as the V4 and V5 series. This can reduce CPU bottlenecks in image pulling. Newer VM SKUs provide a more performant experience across storage and network throughput.

Increase the size of the VM SKU

Increase the available CPU cores on your VM SKU to reduce a CPU bottleneck in image pulling. Allocating more cores helps avoid throttling on the VM.

Toggle image pull type

Use the following AFEC flag instructions to toggle parallel images on or off.

When you run the `az feature list --namespace Microsoft.ContainerService`, you should see the following feature appear:

```
{ "id": "/subscriptions/XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXX/providers/Microsoft.Features/providers/microsoft.ContainerService/features/DisableParallelImagePulls", "name": "
```

```
"microsoft.ContainerService/DisableParallelImagePulls", "properties": { "state": "NotRegistered" }, "type": "Microsoft.Features/providers/features" }
```

You can register the feature on your Azure subscription by the following command:

```
az feature register --namespace Microsoft.ContainerService --name DisableParallelImagePulls
```

Behavior expectation if the feature is registered:

- By default, upgrading any node pool or cluster to Kubernetes 1.31 from a version that's earlier than 1.31 will cause serial image pulling, and upgrading in Kubernetes 1.31 or a later version will cause parallel pulling.
- Node pools or clusters that are already running Kubernetes 1.31 but were created before the subscription registration will initially use parallel image pulling. However, any subsequent node image upgrade or Kubernetes version upgrade will cause the node pool or cluster to use serial image pulling.
- For any new clusters or node pools that are created or upgraded to Kubernetes versions earlier than 1.31, registering for this feature will have no effect. This is because parallel image pulling applies to only versions 1.31 and later.
- Performing an empty `PUT` on existing Kubernetes 1.31 node pools or clusters (that were created before subscription registration) will not immediately switch parallel image pulling to serial pulling. To trigger the change, the node must undergo a reimaging. A reimaging occurs only during a Kubernetes or node image upgrade. However, the `PUT` operation will update the flag in the database, and the change will take effect during the next upgrade.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

Troubleshoot Azure Kubernetes Service clusters or nodes in a failed state

Article • 03/10/2025

This article discusses how to troubleshoot a Microsoft Azure Kubernetes Service (AKS) cluster or node that enters a failed state.

Common causes

Here are the common causes of a failed cluster or node pool:

[+] Expand table

Cause	Reference
Custom Script Extension (CSE) virtual machine (VM) extension provisioning error	Troubleshoot node not ready failures caused by CSE errors
Key Azure resources unavailable	<ul style="list-style-type: none">"Unable to get log analytics workspace info" errorCluster Load Balancer InvalidResourceReference errorSubnetFull errorPrivate DNS Zone InternalOperationError
VM allocation failure due to no zonal/regional capacity	<ul style="list-style-type: none">AllocationFailed or ZonalAllocationFailed errorAssociate capacity reservation groups to node pools
VM allocation failure due to exceeded core quota	Quotaexceeded error
Customer imposed restrictions	<ul style="list-style-type: none">RequestDisallowedByPolicy errorSetting resource lock prevents nodepool from scaling down ↗
Workload issues	<ul style="list-style-type: none">A PodDisruptionBudget (PDB) failed to drain podsThe kube-system pods are not running

Basic troubleshooting of common errors causing a cluster/node to fail

The following table outlines some common errors that can cause a cluster or node to enter a failed state, their descriptions, and basic troubleshooting methods to resolve these errors.

[+] Expand table

Error	Description	Troubleshooting method
OutboundConnFailVMExtensionError	This error indicates that the VM extension fails to install or update due to a lack of outbound connectivity.	Check the network security group (NSG) rules and firewall settings of the VM or VM scale set. Make sure that the VM or VM scale set can access these endpoints: https://aka.ms/aks/outbound , https://aka.ms/aks/ssh , https://aka.ms/aks/agent , and https://aka.ms/aks/containerinsights .
Drain error	This error indicates that the node fails to drain before the upgrade operation.	Check the pod status and events on the node using the kubectl commands: <code>kubectl get pods --all-namespaces -o wide</code> and <code>kubectl describe pod <pod-name> -n <namespace></code> . Look for any pods stuck in a terminated or unknown state or with any errors or warnings in the events. You may need to force delete the pods or restart the kubelet service on the node.
SubscriptionNotRegistered error	This error indicates that the subscription isn't registered to use the AKS resource provider.	Register the subscription using the <code>az provider register --namespace Microsoft.ContainerService</code> command.
RequestDisallowedByPolicy error	This error indicates that the operation is blocked by a policy applied to the	Review the policy details and policy assignment scopes. To allow the operation, you may need to modify or exclude the policy.

Error	Description	Troubleshooting method
	subscription or resource group.	
QuotaExceeded error	This error indicates that the operation exceeds the quota limit for a resource type or a region.	Check the current quota usage and quota limit for the resource type or region using the Azure portal, Azure CLI, or Azure PowerShell. You may need to delete some unused resources or request a quota increase.
PublicIPCountLimitReached error	This error indicates that the operation reaches the maximum number of public IP addresses that can be created in a subscription or region.	Check the current public IP address usage and limit for the subscription or region using the Azure portal, Azure CLI, or Azure PowerShell. You may need to delete some unused public IP addresses or request an increase in the public IP address limit.
OverconstrainedAllocationRequest error	This error indicates that the operation fails to allocate the requested VM size in a region.	Check the availability of the VM size in the region using the Azure portal, Azure CLI, or Azure PowerShell. You may need to choose a different VM size or a different region.
ReadOnlyDisabledSubscription error	This error indicates that the subscription is currently disabled and set to read-only.	Review and adjust the subscription permissions, as the subscription may have been suspended due to billing issues, expired credit, or policy violations.

Basic troubleshooting of other possible issues causing a cluster/node to fail

This table describes other possible issues that can cause a cluster or node to enter a failed state, their descriptions, and solutions to fix these issues.

[+] Expand table

Issue	Description	Solution
The subnet size is too small	The operation can't create or update the cluster because the subnet doesn't have enough space accommodate the required number of nodes.	Delete the node pool and create a new one with a larger subnet size using the Azure portal, Azure CLI, or Azure PowerShell.
The virtual network is blocked	The operation can't communicate with the cluster API server or Kubernetes control plane because the firewall or a custom Domain Name System (DNS) setting blocks the outbound connections from the nodes.	Allow the node's traffic on the firewall and set up the DNS resolution to Azure using the Azure portal, Azure CLI, or Azure PowerShell.
PDB problems	The operation can't update the cluster because a PDB stopped the removal of one or more pods. A PDB is a resource that limits how many pods can be voluntarily terminated during a specific period.	Temporarily remove the PDB, reconcile the cluster, and then add the PDB again using the kubectl command-line tool.
Infrastructure issues	The operation can't update the cluster because of an internal issue with the Azure Resource Manager (ARM) service that manages resources in Azure.	Do an agent pool reconciliation for each node pool and a reconciliation for the managed cluster using the Azure CLI or Azure PowerShell.
API server errors	The operation can't reach the cluster API server or Kubernetes control plane because of an outage or a bug.	Report it to the AKS support team and provide the relevant logs and diagnostic information. You can get the logs and diagnostic information using the Azure portal, Azure CLI, or Azure PowerShell.

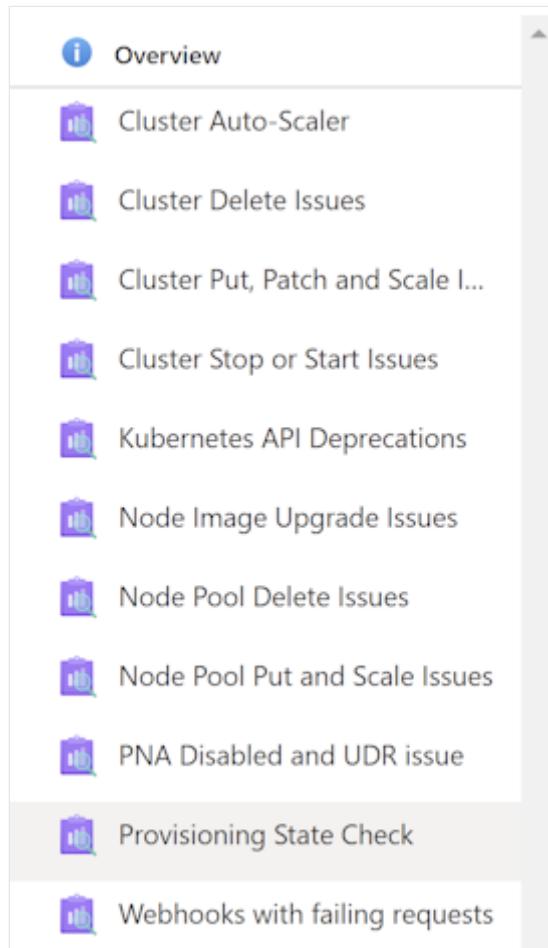
ⓘ Note

- The operation mentioned in the previous table refers to any update (`PUT`) operation triggered from the customer side.
- In Kubernetes, there's a component within a controller. It ensures the actual state of the world, including the cluster state and potentially external states like running containers for Kubelet or load balancers for a cloud provider. It aligns with the desired state specified in an object. This alignment process is a

key function of the controller. For AKS, this component ensures that the state of the AKS cluster aligns with the desired configuration. To trigger it manually, run `az resource update --ids <AKS cluster id>`. You can get the AKS cluster ID by running `az aks show -n <cluster name> -g <cluster resource group> -o json --query id`. If there are any differences between the actual and desired states, take necessary actions to correct these discrepancies.

Provisioning State Check

To check the cluster status, select **Provisioning State Check**. Then, the provisioning state of the cluster and agent pool is shown.



Scenario 1: Cluster is in a failed state

To resolve this issue, get the operation that causes the failure and figure out the error. Here are two common operation failures that can result in a failed cluster:

- Cluster creation failed
- Cluster upgrade failed

If a recently created or upgraded cluster is in a failed state, use the following methods to troubleshoot the failure:

- Examine the [activity log](#) to identify the root cause of the failure.

You can view the activity log using [the Azure portal](#), [Azure CLI](#), or [Azure PowerShell](#).

The activity log can show you the error code and message associated with the failure. For more information about specific errors, see the [Basic troubleshooting of common errors causing a cluster/node to fail](#) section.

- Use the [AKS Diagnose and Solve Problems feature](#) to troubleshoot and resolve common issues.

 **Note**

This feature is only available in the Azure portal and Azure CLI.

View the activity log for a failed cluster using the Azure portal

To view the activity logs for a failed cluster from the Azure portal, follow these steps:

1. In the Azure portal, go to the [Resource groups](#) page and select the resource group that contains your cluster.
2. On the [Overview](#) page, select the cluster name from the resource list.
3. On the cluster page, select **Activity log** from the left menu.
4. On the **Activity log** page, you can filter events by **Status**, **Timespan**, **Event initiated by**, and **Event category**. For example, you can select **Failed** from the **Status** drop-down list to see only failed events.

Resource type	▼
Resource type	
Operation	An action or command, such as create, delete, and write, that affects Azure Resource Manager resources
Event initiated by	The user who started an operation
Event category	The event type for certain operations

5. To check the details of an event, select the event name from the list. A new pane opens with the event summary, properties, and JSON data. You can also download the JSON data as a file.
6. To check the error code and message associated with the event, scroll down to the **Status** section in the event summary. You can also find the error information in the properties and JSON data sections.

View the activity log for a failed cluster using the Azure CLI

If you prefer to use Azure CLI to view the activity log for a failed cluster, follow these steps:

1. Install Azure CLI on your machine and log in with your Azure account.
2. List the resource groups in your subscription using the `az group list` command and find the name of the resource group that contains your cluster.
3. List the resources in the resource group using the `az resource list` command with the `--resource-group` parameter and find the name of the cluster.

4. List the cluster's activity log using the `az monitor activity-log list` command with the `--resource-group` and `--resource` parameters. You can also use the `--status`, `--start-time`, `--end-time`, `--caller`, and `--filter` parameters to filter events by different criteria. For example, you can use `--status Failed` to see only failed events.
5. Show the details of a specific event using the `az monitor activity-log show` command with the `--resource-group`, `--resource`, and `--event-id` parameters. You can find the event ID from the output of the previous command. The output will include the event summary, properties, and JSON data. You can also use the `--output` parameter to change the output format.
6. To see the error code and message associated with the event, look for the `statusMessage` field in the command output. You can also find the error information in the properties and JSON data sections.



```
20 items.  
Operation name  
New recommendation is available  
> Stop Managed Cluster  
Summary JSON Change history (Change history is not applicable to this type of activity log)  
1 {  
2   "authorization": {  
3     "action": "Microsoft.ContainerService/managedClusters/write",  
4     "condition": null,  
5     "role": null.  
6   }  
7 }
```

Use the AKS Diagnose and Solve Problems feature for a failed cluster

In the Azure portal, navigate to your AKS cluster resource and select **Diagnose and solve problems** from the left menu. You'll see a list of categories and scenarios that you can select to run diagnostic checks and get recommended solutions.

In the Azure CLI, use the `az aks kollect` command with the `--name` and `--resource-group` parameters to collect diagnostic data from your cluster nodes. You can also use the `--storage-account` and `--sas-token` parameters to specify an Azure Storage account where the data will be uploaded. The output will include a link to the **Diagnose and Solve Problems** blade where you can view the results and suggested actions.

In the **Diagnose and Solve Problems** blade, you can select **Cluster Issues** as the category. If any issues are detected, you'll see a list of possible solutions that you can follow to fix them.

Scenario 2: Node is in a failed state

In rare cases, an Azure Disk detach operation may partially fail, which leaves the node VM in a failed state.

To resolve this issue, manually update the VM status using one of the following methods:

- For a cluster that's based on an availability set, run the following [az vm update](#) command:

Azure CLI

```
az vm update --resource-group <resource-group-name> --name <vm-name>
```

- For a cluster that's based on a VM scale set, run the following [az vmss update-instances](#) command:

Azure CLI

```
az vmss update-instances --resource-group <resource-group-name> --name <scale-set-name> --instance-id <vm-or-scale-set-id>
```

Scenario 3: Node pool is in a failed state

This issue can happen when the VM scale set or availability set that backs the node pool encounters an error during provisioning, scaling, or updating. This issue can be due to insufficient capacity, quota limits, network issues, policy violations, resource locks, or other factors that prevent the VM from being allocated or configured properly.

To troubleshoot this issue, follow these steps:

1. Check the status of the node pool using the `az aks nodepool show` command. If the provisioning state is `Failed`, you can see the error message and code in the output.
2. Check the status of the VM scale set or availability set using the `az vmss show` or `az vm availability-set show` command. If the provisioning state is `Failed`, you can see the error message and code in the output.
3. Check the status of the individual VM in the node pool using the `az vmss list-instances` or `az vm list` command. If any VM is in a `Failed` or `Unhealthy` state, you can see the error message and code in the output.
4. Check the activity log and diagnostic setting of the VM scale set or availability set to see if any events or alerts that indicate the cause of the failure. You can use the Azure portal, Azure CLI, or Azure Monitor API to access the activity log and diagnostic setting.
5. Check the quota and capacity of the region and subscription where the node pool is deployed. You can use the `az vm list-usage` command or the Azure portal to check the quota and capacity. If the quota or capacity limit is reached, you can request an increase or delete some unused resources.
6. Check the policy and role assignments of the node pool. You can use the `az policy` and `az role` commands or the Azure portal to check the policy definitions, assignments, compliance, and exemptions. You can also check the role assignments and permissions of the node pool using the `az role assignment` command or the Azure portal.
7. Check the resource locks of the node pool. You can use the `az lock` command or the Azure portal to check the lock level, scope, and notes. You can also delete or update the lock if needed.

Other logging and diagnostic tools

If the previous troubleshooting methods don't resolve your issue, you can use the following logging and diagnostic tools to collect more information and identify the root cause:

- Azure Monitor for containers:

This service collects and analyzes metrics and logs from AKS clusters and the nodes. Azure Monitor for containers can monitor cluster and node health, performance, and availability. You can also use it to view container logs, kubelet logs, and node boot diagnostic logs.

- AKS Periscope 

This tool collects node and pod logs, network information, and cluster configuration from an AKS cluster and uploads them to an Azure storage account. This tool can help you troubleshoot common cluster issues, such as DNS resolution, network connectivity, and node status. You can also use it to generate a support request with the collected logs attached.

- AKS Diagnostics

This tool runs a series of checks on AKS clusters and the nodes and provides recommendations and remediation steps for common issues. This tool can help you troubleshoot issues related to cluster creation, upgrade, scaling, networking, storage, and security. You can also use it to generate a support request with the diagnostic results attached.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

Troubleshoot high CPU usage in AKS clusters

Article • 05/21/2025

! Note

This article discusses high CPU utilization. In many situations, CPU Pressure Stall Information (PSI) metrics provide a more accurate indication of CPU Pressure than utilization alone. For more information, see [Troubleshoot CPU pressure in AKS clusters using PSI metrics](#).

High CPU usage is a symptom of one or more applications or processes that require so much CPU time that the performance or usability of the machine is impacted. High CPU usage can occur in many ways, but it's mostly caused by user configuration.

When a node in an [Azure Kubernetes Service \(AKS\)](#) cluster experiences high CPU usage, the applications running on it can experience degradation in performance and reliability. Applications or processes also become unstable, which may lead to issues beyond slow responses.

This article helps you identify the nodes and containers that consume high CPU and provides best practices to resolve high CPU usage.

Symptoms

The following table outlines the common symptoms of high CPU usage:

Expand table

Symptom	Description
CPU starvation	CPU-intensive applications slow down other applications on the same node.
Slow state changes	Pods may take longer to get ready.
NotReady node state	A node enters the NotReady state. This issue occurs because the container with high CPU usage causes the Kubectl command line tool to be unresponsive.

Troubleshooting checklist

To resolve high CPU usage, use effective monitoring tools and apply best practices.

Step 1: identify nodes/containers with high CPU usage

Use either of the following methods to identify nodes and containers with high CPU usage:

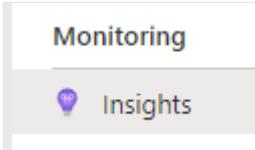
- In a web browser, use the Container Insights feature of AKS in the Azure portal.
- In a console, use the Kubernetes command-line tool (kubectl).

Browser

Container Insights is a feature within AKS. It's designed to monitor the performance of container workloads. You can use Container insights to identify nodes, containers, or pods that drive high CPU usage.

To identify nodes, containers, or pods that drive high CPU usage, follow these steps:

1. Navigate to the cluster from the [Azure portal](#).
2. Under **Monitoring**, select **Insights**.



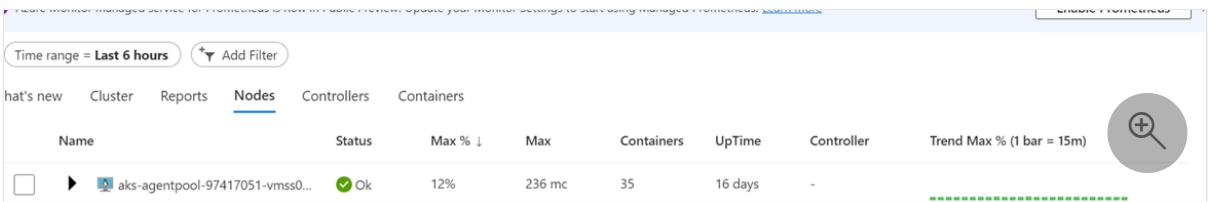
3. Set the appropriate **Time range**.



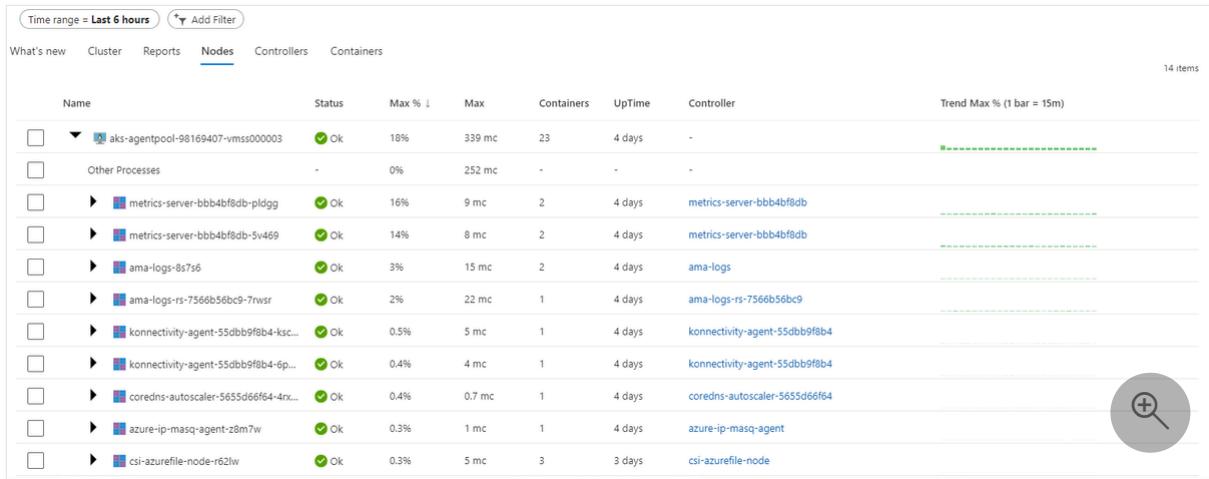
4. Locate the nodes with high CPU usage and check if the node CPU usage is stable.

Select **Nodes**. Set **Metric** to **CPU Usage (millicores)** and then set the sample to **Max**. Use the sort feature on the **Max** to order the nodes by **Max%**. The nodes with the highest CPU usage appear at the top.

In the following screenshot, the node only uses 12% of the max CPU and has been running for 16 days.



5. Once you locate the nodes with high CPU usage, select the nodes to find pods on them and their CPU usage.



① Note

The percentage of CPU or memory usage for pods is based on the CPU request specified for the container. It doesn't represent the percentage of the CPU or memory usage for the node. So, look at the actual CPU or memory usage rather than the percentage of CPU or memory usage for pods.

Once you get the list of pods with high CPU usage, you can map it to the applications that cause the spike in CPU usage.

Step 2: Review best practices to avoid high CPU usage

Review the following table to learn how to implement best practices for avoiding high CPU usage:

Expand table

Best practice	Description
Set appropriate limits for containers	Kubernetes allows specifying requests and limits on the resources for containers. Resource requests and limits represent the minimum and maximum number of resources a container can use. We recommend you set appropriate requests and limits to choose the appropriate Kubernetes Quality of Service (QoS) class for each pod.
Enable Horizontal Pod Autoscaler (HPA)	Setting appropriate limits along with enabling HPA can help in resolving high CPU usage.

Best practice	Description
Select higher SKU VMs	To handle high CPU workloads, use higher SKU VMs. To do this, create a new node pool, cordon off the nodes to make them unschedulable, and drain the existing node pool.
Isolate system and user workloads	We recommend that you create a separate node pool (other than the agent pool) to run your workloads. This can prevent overloading the system node pool and provide better performance.

References

- [Container insights overview](#)
- [Monitor your Kubernetes cluster performance with Container insights](#)
- [How to manage the Container insights agent](#)
- [Resource Limits](#)
- [Resource Quotas](#)
- [Limit Ranges](#)
- [Quality of Service](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Troubleshoot CPU pressure in AKS clusters using PSI metrics

Article • 05/21/2025

CPU pressure is a more accurate indicator of resource contention than traditional CPU utilization metrics. While high CPU usage shows resource consumption, it doesn't necessarily indicate performance problems. In an Azure Kubernetes Service (AKS) cluster, understanding CPU pressure through Pressure Stall Information (PSI) metrics helps identify true resource contention issues.

When a node in an AKS cluster experiences CPU pressure, applications might suffer from poor performance even when CPU utilization appears moderate. PSI metrics provide insight into actual resource contention by measuring task delays rather than just resource consumption.

This article helps you monitor CPU pressure using PSI metrics and provides best practices to resolve resource contention issues.

Symptoms

The following table outlines the common symptoms of CPU pressure:

Expand table

Symptom	Description
Increased application latency	Services respond slower even when CPU utilization appears moderate.
Throttled containers	Containers experience delays in processing despite having CPU resources available on the node.
Degraded performance	Applications experience unpredictable performance variations that don't correlate with CPU usage percentages.

Troubleshooting checklist

To identify and resolve CPU pressure issues, follow these steps:

Step 1: Enable and monitor PSI metrics

Use one of the following methods to access PSI metrics:

- In a web browser, use Azure Monitoring Managed Prometheus or other monitoring solution to query PSI metrics.
- In a console, use the Kubernetes command-line tool (`kubectl`).

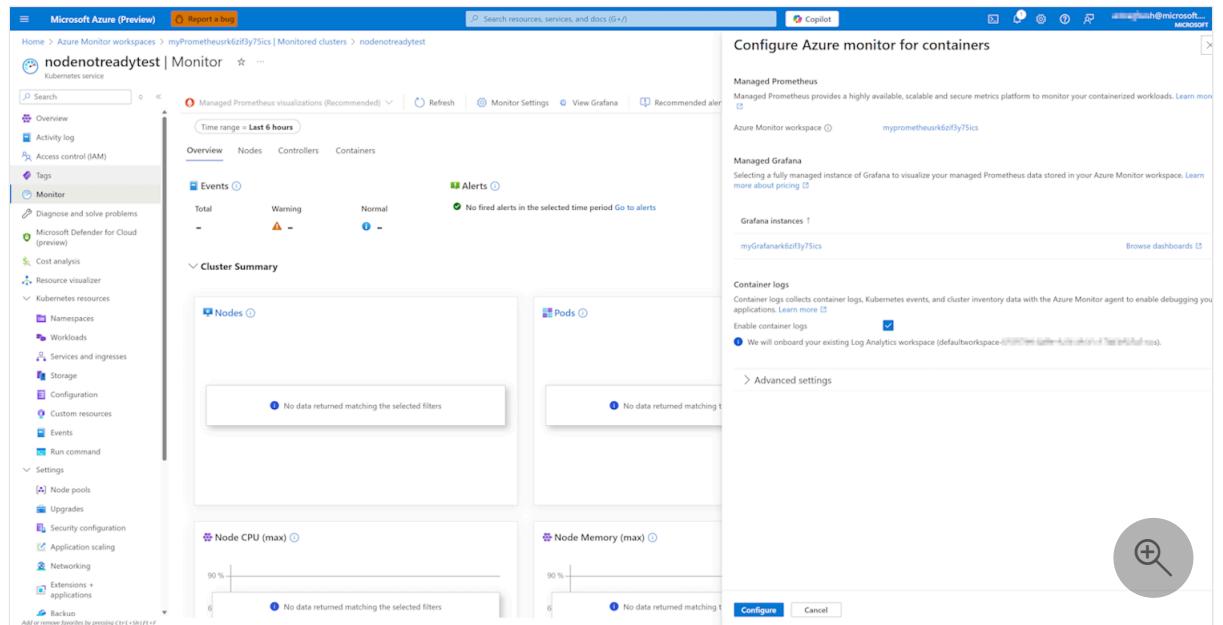


Azure Monitoring Managed Prometheus provides a way to monitor PSI metrics:

1. Enable Azure Monitoring Managed Prometheus for your AKS cluster by following the instructions in [Enable Prometheus and Grafana](#).

To enable customized scrape metrics for Prometheus, see [Scrape configs](#). We recommend setting `minimum ingestion profile` to `false` and `node-exporter` to `true`.

2. Navigate to the Azure Monitor workspace associated with the AKS cluster from the [Azure portal](#).



3. Under **Monitoring**, select **Metrics**.

4. Select **Prometheus metrics** as the data source.

ⓘ Note

To use the metrics, you need to enable them in Azure Monitoring Managed Prometheus. These metrics are exposed by Node Exporter or cAdvisor.

5. Query specific PSI metrics in Prometheus explorer:

- For node-level CPU pressure, use the `node_pressure_cpu_waiting_seconds_total` Prometheus Query Language (PromQL).



- For pod-level CPU pressure, use the `container_cpu_cfs_throttled_seconds_total` PromQL.

6. Calculate the PSI-some percentage (percentage of time at least one task is stalled on CPU):

```
rate(node_pressure_cpu_waiting_seconds_total[5m]) * 100
```

(!) Note

Some of the container level metrics such as

`container_pressure_cpu_waiting_seconds_total` and

`container_pressure_cpu_stalled_seconds_total` aren't available in AKS as they're part of the Kubelet PSI feature gate that is in alpha state. AKS begins supporting the use of the feature when it reaches beta stage.

Step 2: Review best practices to prevent CPU pressure

Review the following table to learn how to implement best practices for avoiding CPU pressure:

[] Expand table

Best practice	Description
Focus on PSI metrics instead of utilization	Use PSI metrics as your primary indicator of resource contention rather than CPU utilization percentages. For more information, see PSI - Pressure Stall Information .

Best practice	Description
Identify pods utilizing the most CPU	Isolate the pods that are utilizing the most CPU and identify solutions to reduce pressure. For more information, see Troubleshoot high CPU usage in AKS clusters .
Minimize CPU limits	Consider removing CPU limits and rely on Linux's Completely Fair Scheduler with CPU shares based on requests. For more information, see Resource Management for Pods and Containers .
Use appropriate Quality of Service (QoS) classes	Set the right QoSClass for each pod based on its importance and contention sensitivity. For more information, see Configure Quality of Service for Pods .
Optimize pod placement	Use pod anti-affinity rules to avoid placing CPU-intensive workloads on the same nodes. For more information, see Assigning Pods to Nodes .
Monitor for brief pressure spikes	Short pressure spikes can indicate issues even when average utilization appears acceptable. For more information, see Resource metrics pipeline .

Key PSI metrics to monitor

! Note

If a node's CPU usage is moderate but the containers on the node experience CFS throttling, increase the resource limits, or remove them and follow [Linux's Completely Fair Scheduler \(CFS\)](#) algorithm.

Node-level PSI metrics

- `node_pressure_cpu_waiting_seconds_total`: Cumulative time tasks wait for CPU.
- `node_cpu_seconds_total`: Traditional CPU utilization for comparison.

Container-level PSI indicators

- `container_cpu_cfs_throttled_periods_total`: The number of periods a container is throttled.
- `container_cpu_cfs_throttled_seconds_total`: Total time a container is throttled.
- Throttling percentage: `rate(container_cpu_cfs_throttled_periods_total[5m]) / rate(container_cpu_cfs_periods_total[5m]) * 100`

Why using PSI metrics?

AKS uses PSI metrics as an indicator for CPU pressure instead of load average for several reasons:

- In oversized and multi-core nodes, load average often underreports CPU saturation.
- On chattier and containerized nodes, load average can over-signal, leading to alert fatigue.
- Since load average doesn't have per-cgroup visibility, noisy pods can hide behind a low system average.

References

- [Linux PSI documentation ↗](#)
- [Kubernetes resource management ↗](#)
- [AKS performance best practices](#)
- [Enable Prometheus and Grafana](#)
- [Quality of Service in Kubernetes ↗](#)
- [Linux Completely Fair Scheduler ↗](#)

Contact us for help

If you have questions or need help, [create a support request ↗](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community ↗](#).

Troubleshoot memory saturation in AKS clusters

06/27/2025

This article discusses methods for troubleshooting memory saturation issues. Memory saturation occurs if at least one application or process needs more memory than a container host can provide, or if the host exhausts its available memory.

Prerequisites

- The Kubernetes [kubectl](#) command-line tool. To install kubectl by using Azure CLI, run the `az aks install-cli` command.
- The open source project [Inspektor Gadget](#) for advanced process level memory analysis. For more information, see [How to install Inspektor Gadget in an AKS cluster](#).

Symptoms

The following table outlines the common symptoms of memory saturation.

Expand table

Symptom	Description
Unschedulable pods	Additional pods can't be scheduled if the node is close to its set memory limit.
Pod eviction	If a node is running out of memory, the kubelet can evict pods. Although the control plane tries to reschedule the evicted pods on other nodes that have resources, there's no guarantee that other nodes have sufficient memory to run these pods.
Node not ready	Memory saturation can cause <code>kubelet</code> and <code>containerd</code> to become unresponsive, eventually causing node readiness issues.
Out-of-memory (OOM) kill	An OOM problem occurs if the pod eviction can't prevent a node issue.

Troubleshooting checklist

To reduce memory saturation, use effective monitoring tools and apply best practices.

Step 1: Identify nodes that have memory saturation

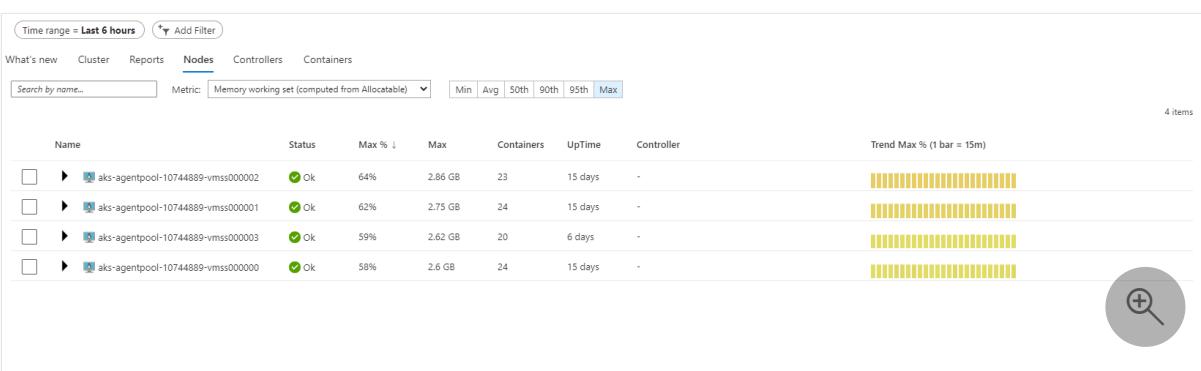
Use either of the following methods to identify nodes that have memory saturation:

- In a web browser, use the Container Insights feature of AKS in the Azure portal.
- In a console, use the Kubernetes command-line tool (kubectl).

Browser

Container Insights is a feature within AKS that monitors container workload performance. For more information, see [Enable Container insights for Azure Kubernetes Service \(AKS\) cluster](#).

1. On the [Azure portal](#), search for and select **Kubernetes services**.
2. In the list of Kubernetes services, select the name of your cluster.
3. In the navigation pane of your cluster, find the **Monitoring** heading, and then select **Insights**.
4. Set the appropriate **Time Range** value.
5. Select the **Nodes** tab.
6. In the **Metric** list, select **Memory working set (computed from Allocatable)**.
7. In the percentiles selector, set the sample to **Max**, and then select the **Max %** column label two times. This action sorts the table nodes by the maximum percentage of memory used, from highest to lowest.



Name	Status	Max %	Max	Containers	UpTime	Controller	Trend Max % (1 bar = 15m)
aks-agentpool-10744889-vmss000002	Ok	64%	2.86 GB	23	15 days	-	
aks-agentpool-10744889-vmss000001	Ok	62%	2.75 GB	24	15 days	-	
aks-agentpool-10744889-vmss000003	Ok	59%	2.62 GB	20	6 days	-	
aks-agentpool-10744889-vmss000000	Ok	58%	2.6 GB	24	15 days	-	

8. Because the first node has the highest memory usage, select that node to investigate the memory usage of the pods that are running on the node.

Time range = Last 6 hours [Add Filter](#)

What's new Cluster Reports **Nodes** Controllers Containers

14 items

Name	Status	Max %	Max	Containers	Uptime	Controller	Trend Max % (1 bar = 15m)
aks-agentpool-98169407-vmss000003	Ok	18%	339 mc	23	4 days	-	
Other Processes	-	0%	252 mc	-	-	-	
metrics-server-bbb4bf8db-pldgg	Ok	16%	9 mc	2	4 days	metrics-server-bbb4bf8db	
metrics-server-bbb4bf8db-5v469	Ok	14%	8 mc	2	4 days	metrics-server-bbb4bf8db	
ama-logs-6s7s6	Ok	3%	15 mc	2	4 days	ama-logs	
ama-logs-rs-7566b56bc9-7rwsr	Ok	2%	22 mc	1	4 days	ama-logs-rs-7566b56bc9	
konnectivity-agent-55dbb9f8b4-ksc...	Ok	0.5%	5 mc	1	4 days	konnectivity-agent-55dbb9f8b4	
konnectivity-agent-55dbb9f8b4-6p...	Ok	0.4%	4 mc	1	4 days	konnectivity-agent-55dbb9f8b4	
coredns-autoscaler-5655d66f64-4rx...	Ok	0.4%	0.7 mc	1	4 days	coredns-autoscaler-5655d66f64	
azure-ip-masq-agent-z8m7w	Ok	0.3%	1 mc	1	4 days	azure-ip-masq-agent	
csi-azurefile-node-r62lw	Ok	0.3%	5 mc	3	3 days	csi-azurefile-node	

! Note

The percentage of CPU or memory usage for pods is based on the CPU request specified for the container. It doesn't represent the percentage of the CPU or memory usage for the node. So, look at the actual CPU or memory usage rather than the percentage of CPU or memory usage for pods.

Now that you've identified the pods that are using high memory, you can identify the applications that are running on the pod or identify processes that may be consuming excess memory.

Step 2: Identify process level memory usage

For advanced process level memory analysis, use [Inspektor Gadget](#) to monitor real time memory usage at the process level within pods:

1. Install Inspektor Gadget using the instructions found in the [documentation](#)
2. Run the [top_process gadget](#) to identify processes that are using large amounts of memory. You can use `--fields` to select certain columns and `--filter` to filter events based on specific field values, for example the pod names of previously identified pods with high memory consumption. You can also:
 - Identify top 10 memory-consuming processes across the cluster:

```
Bash
```

```
kubectl gadget run top_process --sort -memoryRelative --max-entries 10
```

- Identify top memory-consuming processes on a specific node:

```
Bash
```

```
kubectl gadget run top_process --sort -memoryRelative --filter k8s.node==<node-name>
```

- Identify top memory-consuming processes in a specific namespace:

```
Bash
```

```
kubectl gadget run top_process --sort -memoryRelative --filter k8s.namespace==<namespace>
```

- Identify top memory-consuming processes in a specific pod:

```
Bash
```

```
kubectl gadget run top_process --sort -memoryRelative --filter k8s.podName==<pod-name>
```

The output of the Inspektor Gadget `top_process` command resembles the following:

```
Output
```

K8S.NODE	K8S.NAMESPACE	K8S.PODNAME		
PID	COMM	MEMORYVIRTUAL	MEMORYRSS	MEMORYRELATIVE
aks-agentpool-3...901-vmss00001	default	memory-stress		
21676	stress	944 MB	943 MB	5.6
aks-agentpool-3...901-vmss00001	default	memory-stress		
21678	stress	944 MB	943 MB	5.6
aks-agentpool-3...901-vmss00001	default	memory-stress		
21677	stress	944 MB	872 MB	5.2
aks-agentpool-3...901-vmss00001	default	memory-stress		
21679	stress	944 MB	796 MB	4.8

You can use this output to identify the processes that are consuming the most memory on the node. The output can include the node name, namespace, pod name, container name, process ID (PID), command name (COMM), CPU and memory usage, check [the documentation ↗](#) for more details.

Step 3: Review best practices to avoid memory saturation

Review the following table to learn how to implement best practices for avoiding memory saturation.

Best practice	Description
Use memory requests and limits	Kubernetes provides options to specify the minimum memory size (<i>request</i>) and the maximum memory size (<i>limit</i>) for a container. By configuring limits on pods, you can avoid memory pressure on the node. Make sure that the aggregate limits for all pods that are running doesn't exceed the node's available memory. This situation is called <i>overcommitting</i> . The Kubernetes scheduler allocates resources based on set requests and limits through Quality of Service (QoS). Without appropriate limits, the scheduler might schedule too many pods on a single node. This might eventually bring down the node. Additionally, while the kubelet is evicting pods, it prioritizes pods in which the memory usage exceeds their defined requests. We recommend that you set the memory request close to the actual usage.
Enable the horizontal pod autoscaler	By scaling the cluster, you can balance the requests across many pods to prevent memory saturation. This technique can reduce the memory footprint on the specific node.
Use anti-affinity tags	For scenarios in which memory is unbounded by design, you can use node selectors and affinity or anti-affinity tags, which can isolate the workload to specific nodes. By using anti-affinity tags, you can prevent other workloads from scheduling pods on these nodes. This reduces the memory saturation problem.
Choose higher SKU VMs	Virtual machines (VMs) that have more random-access memory (RAM) are better suited to handle high memory usage. To use this option, you must create a new node pool, cordon the nodes (make them unschedulable), and drain the existing node pool.
Isolate system and user workloads	We recommend that you run your applications on a user node pool. This configuration makes sure that you can isolate the Kubernetes-specific pods to the system node pool and maintain the cluster performance.

More information

- [Learn more about Azure Kubernetes Service \(AKS\) best practices](#)
- [Monitor your Kubernetes cluster performance with Container insights](#)

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Troubleshoot high memory consumption in disk-intensive applications

Article • 04/30/2025

Disk input and output operations are costly, and most operating systems implement caching strategies for reading and writing data to the filesystem. The [Linux kernel](#) usually uses strategies such as the [page cache](#) to improve overall performance. The primary goal of the page cache is to store data read from the filesystem in the cache, making it available in memory for future read operations.

This article helps you identify and avoid high memory consumption in disk-intensive applications due to Linux kernel behaviors on Kubernetes pods.

Prerequisites

A tool to connect to the Kubernetes cluster, such as the `kubectl` tool. To install `kubectl` using the [Azure CLI](#), run the `az aks install-cli` command.

Symptoms

When a disk-intensive application running on a pod performs frequent filesystem operations, high memory consumption might occur.

The following table outlines common symptoms of high memory consumption:

Expand table

Symptom	Description
The working set metric is too high.	This issue occurs when there's a significant difference between the working set metric reported by the Kubernetes Metrics API and the actual memory consumed by an application.
Out-of-memory (OOM) kill.	This issue indicates memory issues exist on your pod.
Increased memory usage after heavy disk activity.	After operations such as backups, large file reads/writes, or data imports, memory consumption rises.
Memory usage grows indefinitely.	The pod's memory consumption increases over time without reducing, like a memory leak, even if the application itself isn't leaking memory.

Troubleshooting checklist

Step 1: Inspect the pod working set

To inspect the working set of pods reported by the Kubernetes Metrics API, run the following [kubectl top pods](#) command:

Console

```
$ kubectl top pods -A | grep -i "<DEPLOYMENT_NAME>"  
NAME                      CPU(cores)   MEMORY(bytes)  
my-deployment-fc94b7f98-m9z21   1m          344Mi
```

For detailed steps about how to identify which pod is consuming excessive memory, see [Troubleshoot memory saturation in AKS clusters](#).

Step 2: Inspect pod memory statistics

To inspect the memory statistics of the [cgroups](#) on the pod that's consuming excessive memory, follow these steps:

ⓘ Note

[Cgroups](#) help enforce resource management for pods and containers, including CPU/memory requests and limits for containerized workloads.

1. Connect to the pod:

Console

```
$ kubectl exec <POD_NAME> -it -- bash
```

2. Navigate to the `cgroup` statistics directory and list the memory-related files:

Console

```
$ ls /sys/fs/cgroup | grep -e memory.stat -e memory.current  
memory.current memory.stat
```

- `memory.current`: Total memory currently used by the `cgroup` and its descendants.

- `memory.stat`: This breaks down the cgroup's memory footprint into different types of memory, type-specific details, and other information about the state and past events of the memory management system.

All the values listed in those files are in bytes.

3. Get an overview of how memory consumption is distributed on the pod:

```
Console

$ cat /sys/fs/cgroup/memory.current
10645012480
$ cat /sys/fs/cgroup/memory.stat
anon 5197824
inactive_anon 5152768
active_anon 8192
...
file 10256240640
active_file 32768
inactive_file 10256207872
...
slab 354682456
slab_reclaimable 354554400
slab_unreclaimable 128056
...
```

`cAdvisor` uses `memory.current` and `inactive_file` to compute the working set metric.

You can replicate the calculation using the following formula:

```
working_set = (memory.current - inactive_file) / 1048576 = (10645012480 -
10256207872) / 1048576 = 370 MB
```

Step 3: Determine kernel and application memory consumption

The following table describes some memory segments:

[] Expand table

Segment	Description
<code>anon</code>	The amount of memory used in anonymous mappings. Most languages use this segment to allocate memory.
<code>file</code>	The amount of memory used to cache filesystem data, including tmpfs and shared memory.
<code>slab</code>	The amount of memory used to store data structures in the Linux kernel.

Combined with Step 2, `anon` represents 5,197,824 bytes, which isn't close to the total amount reported by the working set metric. The `slab` memory segment used by the Linux kernel represents 354,682,456 bytes, which is almost all the memory reported by the working set metric on the pod.

Step 4: Drop the kernel cache on a debugger pod

! Note

This step might lead to availability and performance issues. Avoid running it in a production environment.

1. Get the node running the pod:

```
Console

$ kubectl get pod -A -o wide | grep "<POD_NAME>"
NAME                  READY   STATUS    RESTARTS   AGE     IP
NODE                 NOMINATED NODE   READINESS GATES
my-deployment-fc94b7f98-m9z21   1/1     Running   0        37m
10.244.1.17   aks-agentpool-26052128-vmss000004   <none>      <none>
```

2. Create a debugger pod using the `kubectl debug` command and create a `kubectl` session:

```
Console

$ kubectl debug node/<NODE_NAME> -it --image=mcr.microsoft.com/cbl-mariner/busybox:2.0
$ chroot /host
```

3. Drop the kernel cache:

```
Console

echo 1 > /proc/sys/vm/drop_caches
```

4. Verify if the command in the previous step causes any effect by repeating Step 1 and Step 2:

```
Console
```

```
$ kubectl top pods -A | grep -i "<DEPLOYMENT_NAME>"  
NAME                                CPU(cores)   MEMORY(bytes)  
my-deployment-fc94b7f98-m9z21      1m           4Mi  
  
$ kubectl exec <POD_NAME> -it -- cat /sys/fs/cgroup/memory.stat  
anon 4632576  
file 1781760  
...  
slab_reclaimable 219312  
slab_unreclaimable 173456  
slab 392768
```

If you observe a significant decrease in both the working set and the `slab` memory segment, you're experiencing an issue with the Linux kernel using a great amount of memory on the pod.

Workaround: Configure appropriate memory limits and requests

The only effective workaround for high memory consumption on Kubernetes pods is to set realistic resource [limits and requests](#). For example:

YAML

```
resources:  
  requests:  
    memory: 30Mi  
  limits:  
    memory: 60Mi
```

By configuring appropriate memory limits and requests in Kubernetes or the specification, you can ensure that Kubernetes manages memory allocation more efficiently, mitigating the impact of excessive kernel-level caching on pod memory usage.

✖ Caution

Misconfigured pod memory limits can lead to problems such as OOMKilled errors.

References

- [Learn more about Azure Kubernetes Service \(AKS\) best practices](#)
- [Monitor your Kubernetes cluster performance with Container insights](#)

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Troubleshoot issues when enabling the AKS cluster service health probe mode

07/16/2025

The health probe mode feature allows you to configure how Azure Load Balancer probes the health of the nodes in your Azure Kubernetes Service (AKS) cluster. You can choose between two modes: Shared and ServiceNodePort. The Shared mode uses a single health probe for all external traffic policy cluster services that use the same load balancer. In contrast, the ServiceNodePort mode uses a separate health probe for each service. The Shared mode can reduce the number of health probes and improve the performance of the load balancer, but it requires some additional components to work properly. To enable this feature, see [How to enable the health probe mode feature using the Azure CLI](#).

This article describes some common issues about using the health probe mode feature in an AKS cluster and helps you troubleshoot and resolve these issues.

Symptoms

When creating or updating an AKS cluster by using the Azure CLI, if you enable the health probe mode feature using the `--cluster-service-load-balancer-health-probe-mode Shared` flag, the following issues occur:

- The load balancer doesn't distribute traffic to the nodes as expected.
- The load balancer reports unhealthy nodes even if they're healthy.
- The health-probe-proxy sidecar container crashes or doesn't start.
- The cloud-node-manager pod crashes or doesn't start.

The following operations also happen:

1. RP frontend checks if the request is valid and updates the corresponding property in the LoadBalancerProfile.
2. RP async calls the cloud provider config secret reconciler to update the cloud provider config secret based on the LoadBalancerProfile.
3. Overlaymgr reconciles the cloud-node-manager chart to enable the health-probe-proxy sidecar.

Initial troubleshooting

To troubleshoot these issues, follow these steps:

1. First, connect to your AKS cluster using the Azure CLI:

Azure CLI

```
export RESOURCE_GROUP="aks-rg"
export AKS_CLUSTER_NAME="aks-cluster"
az aks get-credentials --resource-group $RESOURCE_GROUP --name
$AKS_CLUSTER_NAME --overwrite-existing
```

2. Next, check the RP frontend log to see if the health probe mode in the LoadBalancerProfile is properly configured. You can use the `az aks show` command to view the LoadBalancerProfile property of your cluster.

Azure CLI

```
export RESOURCE_GROUP="aks-rg"
export AKS_CLUSTER_NAME="aks-cluster"
az aks show --resource-group $RESOURCE_GROUP --name $AKS_CLUSTER_NAME --query
"networkProfile.loadBalancerProfile"
```

Results:

Output

```
{
  "clusterServiceLoadBalancerHealthProbeMode": "Shared",
  "managedOutboundIPs": null,
  "outboundIPs": null,
  "outboundIPPrefixes": null,
  "allocatedOutboundPorts": null,
  "effectiveOutboundIPs": [
    {
      "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxx/resourceGroups/MC_aks-rg_aks-
cluster_eastus2/providers/Microsoft.Network/publicIPAddresses/xxxxxxxxxxxxxx
xxxxxxxxxxxxxxx"
    }
  ],
  "idleTimeoutInMinutes": 30,
  "loadBalancerSku": "standard",
  "managedOutboundIPv6": null
}
```

3. Check the cloud provider configuration. In modern AKS clusters, the cloud provider configuration is managed internally and the `ccp` namespace doesn't exist. Instead, check for cloud provider related resources and verify the cloud-node-manager pods are running properly:

Bash

```
# Check for cloud provider related ConfigMaps in kube-system
kubectl get configmap -n kube-system | grep -i azure

# Check if cloud-node-manager pods are running (indicates cloud provider
# integration is working)
kubectl get pods -n kube-system | grep cloud-node-manager

# Check the azure-ip-masq-agent-config if it exists
kubectl get configmap azure-ip-masq-agent-config-reconciled -n kube-system -o
yaml 2>/dev/null || echo "ConfigMap not found"
```

Results:

Output

configmap/azure-ip-masq-agent-config-reconciled	1	11h
cloud-node-manager-rfb2w	2/2	Running 0
16m		

4. Check the chart or overlay daemonset cloud-node-manager to see if the health-probe-proxy sidecar container is enabled. You can use the `kubectl get ds` command to view the daemonset.

shell

```
kubectl get ds -n kube-system cloud-node-manager -o yaml
```

Results:

Output

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: cloud-node-manager
  namespace: kube-system
  ...
spec:
  template:
```

```
spec:  
  containers:  
    - name: cloud-node-manager  
      image: mcr.microsoft.com/oss/kubernetes/azure-cloud-node-  
manager:xxxxxxxx  
    - name: health-probe-proxy  
      image: mcr.microsoft.com/oss/kubernetes/azure-health-probe-  
proxy:xxxxxxxx  
    ...
```

Cause 1: The health probe mode isn't Shared or ServiceNodePort

The health probe mode feature only works with these two modes. If you use any other mode, the feature won't work.

Solution 1: Use the correct health probe mode

Make sure you use the Shared or ServiceNodePort mode when creating or updating your cluster. You can use the `--cluster-service-load-balancer-health-probe-mode` flag to specify the mode.

Cause 2: The toggle for the health probe mode feature is off

The health probe mode feature is controlled by a toggle that can be enabled or disabled by the AKS team. If the toggle is off, the feature won't work.

Solution 2: Turn on the toggle

Contact the AKS team to check if the toggle for the health probe mode feature is on or off. If it's off, ask them to turn it on for your subscription.

Cause 3: The load balancer SKU is Basic

The health probe mode feature only works with the Standard Load Balancer SKU. If you use the Basic Load Balancer SKU, the feature won't work.

Solution 3: Use the Standard Load Balancer SKU

Make sure you use the Standard Load Balancer SKU when creating or updating your cluster. You can use the `--load-balancer-sku` flag to specify the SKU.

Cause 4: The feature isn't registered

The health probe mode feature requires you to register the feature on your subscription. If the feature isn't registered, it won't work.

Solution 4: Register the feature

Make sure you register the feature for your subscription before creating or updating your cluster. You can use the `az feature register` command to register the feature.

Azure CLI

```
export FEATURE_NAME="EnableSLBSharedHealthProbePreview"
export PROVIDER_NAMESPACE="Microsoft.ContainerService"
az feature register --name $FEATURE_NAME --namespace $PROVIDER_NAMESPACE
```

Results:

Output

```
{
  "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxx/providers/Microsoft.Features/providers/Microsoft.ContainerService/fea-
tures/EnableAKSClusterServiceLoadBalancerHealthProbeMode",
  "name": "Microsoft.ContainerService/EnableAKSClusterServiceLoadBalancerHealthProbeMode",
  "properties": {
    "state": "Registering"
  },
  "type": "Microsoft.Features/providers/features"
}
```

Cause 5: The Kubernetes version is earlier than v1.28.0

The health probe mode feature requires a minimum Kubernetes version of v1.28.0. If you use an older version, the feature won't work.

Solution 5: Upgrade the Kubernetes version

Make sure you use Kubernetes v1.28.0 or a later version when creating or updating your cluster. You can use the `--kubernetes-version` flag to specify the version.

Known issues

For Windows, the kube-proxy component doesn't start until you create the first non-HPC pod in a node. This issue affects the health probe mode feature and causes the load balancer to report unhealthy nodes. It will be fixed in a future update.

How to enable the health probe mode feature using the Azure CLI

To enable the health probe mode feature, run one of the following commands:

Enable `ServiceNodePort` health probe mode (default) for a cluster:

```
shell

export RESOURCE_GROUP="aks-rg"
export AKS_CLUSTER_NAME="aks-cluster"
az aks update --resource-group $RESOURCE_GROUP --name $AKS_CLUSTER_NAME --cluster-service-load-balancer-health-probe-mode ServiceNodePort
```

Results:

```
Output

{
  "name": "aks-cluster",
  "location": "eastus2",
  "resourceGroup": "aks-rg",
  "kubernetesVersion": "1.28.x",
  "provisioningState": "Succeeded",
  "loadBalancerProfile": {
    "clusterServiceLoadBalancerHealthProbeMode": "ServiceNodePort",
    ...
  },
  ...
}
```

Enable `Shared` health probe mode for a cluster:

```
shell
```

```
export RESOURCE_GROUP="MyAksResourceGroup"
export AKS_CLUSTER_NAME="MyAksCluster"
az aks update --resource-group $RESOURCE_GROUP --name $AKS_CLUSTER_NAME --cluster-service-load-balancer-health-probe-mode Shared
```

Results:

Output

```
{
  "name": "MyAksCluster",
  "location": "eastus2",
  "resourceGroup": "MyAksResourceGroup",
  "kubernetesVersion": "1.28.x",
  "provisioningState": "Succeeded",
  "loadBalancerProfile": {
    "clusterServiceLoadBalancerHealthProbeMode": "Shared",
    ...
  },
  ...
}
```

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Troubleshoot pod scheduler errors in Azure Kubernetes Service

07/02/2025

When you deploy workloads in Azure Kubernetes Service (AKS), you might encounter scheduler errors that prevent Pods from running. This article provides solutions to common scheduler errors.

Error: 0/(X) nodes are available: Y node(s) had volume node affinity conflict

! Note

X and Y represent the number of nodes. These values depend on your cluster configuration.

Pods remain in the Pending state with the following scheduler error:

0/(X) nodes are available: Y node(s) had volume node affinity conflict.

Cause

Persistent Volumes [🔗](#) define `nodeAffinity` rules that restrict which nodes can access the volume. If none of the available nodes satisfy the volume's affinity rules, the scheduler cannot assign the Pod to any node.

Solution

1. Review the node affinity set on your Persistent Volume resource:

Bash

```
kubectl get pv <pv-name> -o yaml
```

2. Check node labels in the cluster:

Bash

```
kubectl get nodes --show-labels
```

3. Make sure that at least one node's labels match the `nodeAffinity` specified in the Persistent Volume's YAML spec.
4. To resolve the conflict, Update the Persistent Volume's `nodeAffinity` rules to match existing node labels or add the required labels to the correct node:

Bash

```
kubectl label nodes <node-name> <key>=<value>
```

Or, modify the PV's affinity rules to match existing node labels.

5. After resolving the conflict, monitor the Pod status or retry the deployment.

Error: 0/(X) nodes are available: Insufficient CPU

Pods remain in the Pending state with the scheduler error:

Error: 0/(X) nodes are available: Insufficient CPU.

Cause

This issue occurs when one or more of the following conditions are met:

- All node resources are in use.
- The pending Pod's resource requests exceed available CPU on the nodes.
- The node pools lack sufficient resources or have incorrect configuration settings.

Solution

1. Review CPU usage on all nodes and verify if there is enough unallocated CPU to meet the pod's request.

Bash

```
kubectl describe pod <pod-name>
kubectl describe nodes
```

2. If no node has enough CPU, increase the number of nodes or use larger VM sizes in the node pool:

```
Bash
```

```
az aks nodepool scale \  
  --resource-group <resource-group> \  
  --cluster-name <aks-name> \  
  --name <nodepool-name> \  
  --node-count <desired-node-count>
```

3. Optimize Pod resource requests. Make sure that CPU requests and limits are appropriate for your node sizes.
4. Verify if any scheduling constraints are restricting pod placement across available nodes.

Error: 0/(X) nodes are available: Y node(s) had untolerated taint

Pods remain in the Pending state with the error:

```
Error: 0/(X) nodes are available: Y node(s) had untolerated taint.
```

Cause

The Kubernetes scheduler tries to assign the Pod to a node, but all nodes are rejected for one of the following reasons:

- The node has a taint (`key=value:effect`) that the Pod doesn't tolerate.
- The node has other taint-based restrictions that prevent the Pod from being scheduled.

Solution

1. Check node taints:

```
Bash
```

```
kubectl get nodes -o json | jq '.items[].spec.taints'
```

2. Add necessary tolerations to Pod spec: Edit your deployment or Pod YAML to include matching tolerations for the taints on your nodes. For example, if your node has the taint

key=value:NoSchedule, your Pod spec must include:

```
yml

tolerations:
- key: "key"
  operator: "Equal"
  value: "value"
  effect: "NoSchedule"
```

If the taint isn't needed, you can remove it from the node:

```
Bash

kubectl taint nodes <node-name> <key>:<effect>-
```

3. Redeploy or monitor the Pod status:

```
Bash

kubectl get pods -o wide
```

Reference

- [Kubernetes: Use Azure Disks with Azure Kubernetes Service](#)
- [Kubernetes: Use node taints](#)
- [Kubernetes Documentation: Insufficient CPU ↗](#)
- [Kubernetes Documentation: Assign and Schedule Pods with Taints and Tolerations ↗](#)

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Contact us for help

If you have questions or need help, [create a support request↗](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community ↗](#).

Basic troubleshooting of Node Not Ready failures

Article • 09/26/2024

This article provides troubleshooting steps to recover Microsoft Azure Kubernetes Service (AKS) cluster nodes after a failure. This article specifically addresses the most common error messages that are generated when a Node Not Ready failure occurs, and explains how node repair functionality can be done for both Windows and Linux nodes.

Before you begin

Read the [official guide for troubleshooting Kubernetes clusters](#). Also, read the [Microsoft engineer's guide to Kubernetes troubleshooting](#). This guide contains commands for troubleshooting pods, nodes, clusters, and other features.

Prerequisites

- [Azure CLI](#), version 2.31 or a later version. If Azure CLI is already installed, you can find the version number by running `az --version`.

Basic troubleshooting

AKS continuously monitors the health state of worker nodes, and [automatically repairs](#) the nodes if they become unhealthy. The Azure Virtual Machine (VM) platform [maintains VMs](#) that experience issues. AKS and Azure VMs work together to reduce service disruptions for clusters.

For nodes, there are two forms of heartbeats:

- Updates to the `.status` of a `Node` object.
- [Lease](#) objects within the [kube-node-lease](#) namespace. Each `Node` has an associated `Lease` object.

Compared to updates to the `.status` of a `Node`, a `Lease` is a lightweight resource. Using `Lease` objects for heartbeats reduces the performance impact of these updates for large clusters.

The [kubelet](#) is responsible for creating and updating the `.status` for `Node` objects. It's also responsible for updating the `Lease` objects that are related to the `Node` objects.

- The kubelet updates the node `.status` when there's a change in status or if there has been no update for a configured interval. The default interval for `.status` updates to nodes is five minutes, which is much longer than the 40-second default time-out for unreachable nodes.
- The kubelet creates and then updates its `Lease` object every 10 seconds (the default update interval). `Lease` updates occur independently from updates to the node `.status`. If the `Lease` update fails, the kubelet retries, using an exponential backoff that starts at 200 milliseconds and is capped at seven seconds.

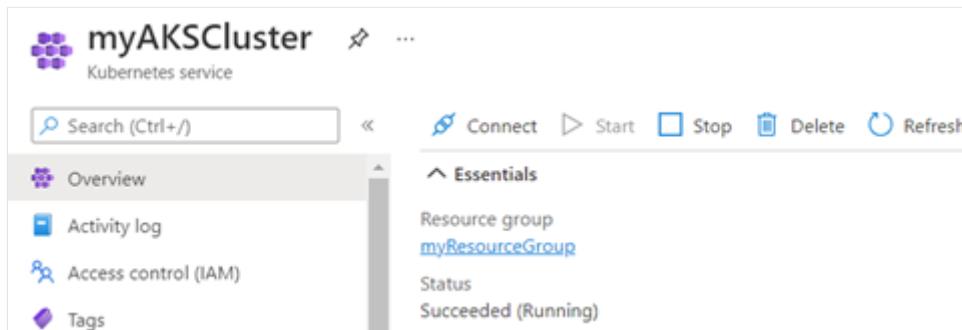
You can't schedule a pod on a node that has a status of `NotReady` or `Unknown`. You can schedule a pod only on nodes that are in the `Ready` state.

If your node is in the `MemoryPressure`, `DiskPressure`, or `PIDPressure` state, you must manage your resources in order to schedule extra pods on the node. If your node is in `NetworkUnavailable` mode, you must configure the network on the node correctly.

AKS manages the lifecycle and operations of agent nodes for you. Modifying the IaaS resources associated with the agent nodes isn't supported. For example, customizing a node through SSH connections, updating packages, or changing the network configuration on a node isn't supported. For more information, see [AKS support coverage for agent nodes](#).

Make sure that the following conditions are met:

- Your cluster is in **Succeeded (Running)** state. To check the cluster status on the [Azure portal](#), search for and select **Kubernetes services**, and select the name of your AKS cluster. Then, on the cluster's **Overview** page, look in **Essentials** to find the **Status**. Or, enter the `az aks show` command in Azure CLI.



- Your node pool has a **Provisioning state** of **Succeeded** and a **Power state** of **Running**. To check the node pool status on the Azure portal, return to your AKS

cluster's page, and then select **Node pools**. Alternatively, enter the `az aks nodepool show` command in Azure CLI.

The screenshot shows the 'Node pools' tab selected in the Azure portal. A summary text states: 'Node pools provide a space for applications to run. Node pools of different sizes can be used to handle a variety of workloads, existing node pools can be scaled and updated, and node pools that are no longer needed can be deleted. Each node pool will contain nodes backed by a Kubernetes node pool.' Below this is a table with three columns: 'Node pool', 'Provisioning state', and 'Power state'. The single row in the table is for 'nodepool1', which has a provisioning state of 'Succeeded' and a power state of 'Running'.

Node pool	Provisioning state	Power state
nodepool1	Succeeded	Running

- The required egress ports are open in your network security groups (NSGs) and firewall so that the API server's IP address can be reached. For more information, see [Required outbound network rules and FQDNs for AKS clusters](#).
- Your nodes [have deployed the latest node images](#).
- Your nodes are in the `Running` state instead of `Stopped` or `Deallocated`.
- Your cluster is running an [AKS-supported version of Kubernetes](#).

More information

To troubleshoot the `Not Ready` status of a node, see [Troubleshoot a change in a healthy node to Not Ready status](#).

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Feedback

Was this page helpful?

Yes

No

Troubleshoot node not ready failures caused by CSE errors

07/09/2025

This article helps you troubleshoot scenarios in which a Microsoft Azure Kubernetes Service (AKS) cluster isn't in the `Succeeded` state and an AKS node isn't ready within a node pool because of custom script extension (CSE) errors.

Prerequisites

- Azure CLI

Symptoms

Because of CSE errors, an AKS cluster node isn't ready within a node pool, and the AKS cluster isn't in the `Succeeded` state.

Cause

The node extension deployment fails and returns more than one error code when you provision the `kubelet` and other components. This is the most common cause of errors. To verify that the node extension deployment is failing when you provision the `kubelet`, follow these steps:

1. To better understand the current failure on the cluster, run the `az aks show` and `az resource update` commands to set up debugging:

Set your environment variables and run the commands to view the cluster's status and debug information.

Azure CLI

```
export RG_NAME="my-aks-rg"
export CLUSTER_NAME="myakscluster"
clusterResourceId=$(az aks show \
    --resource-group $RG_NAME --name $CLUSTER_NAME --output tsv --query id)
az resource update --debug --verbose --ids $clusterResourceId
```

Results:

Output

```
{  
  "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx/resourceGroups/my-aks-rg-  
  xxx/providers/Microsoft.ContainerService/managedClusters/myaksclusterxxx",  
  "name": "myaksclusterxxx",  
  "type": "Microsoft.ContainerService/managedClusters",  
  "location": "eastus2",  
  "tags": null,  
  "properties": {  
    ...  
  }  
}
```

2. Check the debugging output and the error messages that you received from the `az resource update` command against the error list in the [CSE helper](#) executable file on GitHub.

If any of the errors involve the CSE deployment of the kubelet, then you've verified that the scenario that's described here's the cause of the Node Not Ready failure.

In general, exit codes identify the specific issue that's causing the failure. For example, you see messages such as "Unable to communicate with API server" or "Unable to connect to internet." Or the exit codes might alert you to API network time-outs, or a node fault that needs a replacement.

Solution 1: Make sure your custom DNS server is configured correctly

Set up your custom Domain Name System (DNS) server so that it can do name resolution correctly. Configure the server to meet the following requirements:

- If you're using custom DNS servers, make sure that the servers are healthy and reachable over the network.
- Make sure that custom DNS servers have the required [conditional forwarders to the Azure DNS IP address](#) (or the forwarder to that address).
- Make sure that your private AKS DNS zone is linked to your custom DNS virtual networks if they're hosted on Azure.
- Don't use the Azure DNS IP address with the IP addresses of your custom DNS server. Doing this isn't recommended.

- Avoid using IP addresses instead of the DNS server in DNS settings. You can use Azure CLI commands to check for this situation on a Virtual Machine Scale Set or availability set.
 - For Virtual Machine Scale Set nodes, use the `az vmss run-command invoke` command:

Important: You must specify the `--instance-id` of the VM scale set. Here, we demonstrate querying for a valid instance ID (e.g., 0) and a likely VMSS in an AKS node resource group. Update values appropriately to match your environment.

Azure CLI

```
export NODE_RESOURCE_GROUP=$(az aks show --resource-group $RG_NAME --name $CLUSTER_NAME --query nodeResourceGroup -o tsv)
export VMSS_NAME=$(az vmss list --resource-group $NODE_RESOURCE_GROUP --query "[0].name" -o tsv)
export DNS_IP_ADDRESS="10.0.0.10"
export INSTANCE_ID=$(az vmss list-instances --resource-group $NODE_RESOURCE_GROUP --name $VMSS_NAME --query "[0].instanceId" -o tsv)
export API_FQDN=$(az aks show --resource-group $RG_NAME --name $CLUSTER_NAME --query fqdn -o tsv)

az vmss run-command invoke \
  --resource-group $NODE_RESOURCE_GROUP \
  --name $VMSS_NAME \
  --instance-id $INSTANCE_ID \
  --command-id RunShellScript \
  --output tsv \
  --query "value[0].message" \
  --scripts "telnet $DNS_IP_ADDRESS 53"
az vmss run-command invoke \
  --resource-group $NODE_RESOURCE_GROUP \
  --name $VMSS_NAME \
  --instance-id $INSTANCE_ID \
  --command-id RunShellScript \
  --output tsv \
  --query "value[0].message" \
  --scripts "nslookup $API_FQDN $DNS_IP_ADDRESS"
```

- For VM availability set nodes, use the `az vm run-command invoke` command:

Important: You must specify the `--name` of a valid VM in an availability set in your resource group. Here is a template for running network checks.

Azure CLI

```
az vm run-command invoke \
  --resource-group $RG_NAME \
  --name $AVAILABILITY_SET_VM \
```

```
--command-id RunShellScript \
--output tsv \
--query "value[0].message" \
--scripts "telnet $DNS_IP_ADDRESS 53"
az vm run-command invoke \
--resource-group $RG_NAME \
--name $AVAILABILITY_SET_VM \
--command-id RunShellScript \
--output tsv \
--query "value[0].message" \
--scripts "nslookup $API_FQDN $DNS_IP_ADDRESS"
```

For more information, see [Name resolution for resources in Azure virtual networks](#) and [Hub and spoke with custom DNS](#).

Solution 2: Fix API network time-outs

Make sure that the API server can be reached and isn't subject to delays. To do this, follow these steps:

- Check the AKS subnet to see whether the assigned network security group (NSG) is blocking the egress traffic port 443 to the API server.
- Check the node itself to see whether the node has another NSG that's blocking the traffic.
- Check the AKS subnet for any assigned route table. If a route table has a network virtual appliance (NVA) or firewall, make sure that port 443 is available for egress traffic. For more information, see [Control egress traffic for cluster nodes in AKS](#).
- If the DNS resolves names successfully and the API is reachable, but the node CSE failed because of an API time-out, take the appropriate action as shown in the following table.

[] [Expand table](#)

Set type	Action
VM availability set	Delete the node from the Azure portal and the AKS API by using the kubectl delete node command, and then scale up the cluster again.
Virtual Machine Scale Set	Either reimagine the node from the Azure portal, or delete the node, and then scale up the cluster again. To delete the specific node, use az aks nodepool delete-machines command. It will cordon & drain first and then delete the node.

- If the requests are being throttled by the AKS API server, upgrade to a higher service tier. For more information, see [Pricing tiers for AKS](#).

More information

- For general troubleshooting steps, see [Basic troubleshooting of Node Not Ready failures](#).

Troubleshoot Node Not Ready failures if there are expired certificates

Article • 04/10/2025

This article helps you troubleshoot Node Not Ready scenarios within a Microsoft Azure Kubernetes Service (AKS) cluster if there are expired certificates.

Prerequisites

- [Azure CLI](#)
- The [OpenSSL](#) command-line tool for certificate display and signing

Symptoms

You discover that an AKS cluster node is in the Node Not Ready state.

Cause

There are one or more expired certificates.

Prevention: Run OpenSSL to sign the certificates

Check the expiration dates of certificates by invoking the [openssl-x509](#) command, as follows:

- For virtual machine (VM) scale set nodes, use the [az vmss run-command invoke](#) command:

Azure CLI

```
az vmss run-command invoke \
    --resource-group <resource-group-name> \
    --name <vm-scale-set-name> \
    --command-id RunShellScript \
    --instance-id 0 \
    --output tsv \
    --query "value[0].message" \
    --scripts "openssl x509 -in /etc/kubernetes/certs/apiserver.crt -noout -enddate"
```

- For VM availability set nodes, use the [az vm run-command invoke](#) command:

Azure CLI

```
az vm run-command invoke \
    --resource-group <resource-group-name> \
    --name <vm-availability-set-name> \
    --command-id RunShellScript \
    --output tsv \
    --query "value[0].message" \
    --scripts "openssl x509 -in /etc/kubernetes/certs/apiserver.crt -noout -enddate"
```

You might receive certain error codes after you invoke these commands. For information about error codes 50, 51, and 52, see the following links, as necessary:

- [Troubleshoot the OutboundConnFailVMExtensionError error code \(50\)](#)
- [Troubleshoot the K8SAPIServerConnFailVMExtensionError error code \(51\)](#)
- [Troubleshoot the K8SAPIServerDNSLookupFailVMExtensionError error code \(52\)](#)

If you receive error code 99, this indicates that the [apt-get update](#) command is being blocked from accessing one or more of the following domains:

- security.ubuntu.com
- azure.archive.ubuntu.com
- nvidia.github.io

To allow access to these domains, update the configuration of any blocking firewalls, network security groups (NSGs), or network virtual appliances (NVAs).

Solution: Rotate the certificates

You can apply [certificate auto rotation](#) to rotate certificates in the nodes before they expire. This option requires no downtime for the AKS cluster.

If you can accommodate cluster downtime, you can [manually rotate the certificates](#) instead.

(!) Note

Starting in the [July 15, 2021, release of AKS](#), an AKS cluster upgrade automatically helps to rotate the cluster certificates. However, this behavioral change doesn't take effect for an expired cluster certificate. If an upgrade takes only the following actions, the expired certificates won't be renewed:

- Upgrade a node image.
- Upgrade a node pool to the same version.

- Upgrade a node pool to a more recent version.

Only a full upgrade (that is, an upgrade for both the control plane and the node pool) helps renew the expired certificates.

More information

- For general troubleshooting steps, see [Basic troubleshooting of Node Not Ready failures](#).

Troubleshoot a change in a healthy node to Not Ready status

Article • 09/05/2024

This article discusses a scenario in which the status of an Azure Kubernetes Service (AKS) cluster node changes to **Not Ready** after the node is in a healthy state for some time. This article outlines the particular cause and provides a possible solution.

Prerequisites

- The Kubernetes [kubectl](#) tool. To install kubectl by using Azure CLI, run the [az aks install-cli](#) command.
- The Kubernetes [kubelet](#) tool.
- The Kubernetes [containerd](#) tool.
- The following Linux tools:
 - [awk](#)
 - [head](#)
 - [journalctl](#)
 - [ps](#)
 - [sort](#)
 - [watch](#)

Symptoms

The status of a cluster node that has a healthy state (all services running) unexpectedly changes to **Not Ready**. To view the status of a node, run the following [kubectl describe](#) command:

```
Bash
```

```
kubectl describe nodes
```

Cause

The [kubelet](#) stopped posting its **Ready** status.

Examine the output of the `kubectl describe nodes` command to find the [Conditions](#) field and the [Capacity and Allocatable](#) blocks. Do the content of these fields appear as

expected? (For example, in the **Conditions** field, does the `message` property contain the "kubelet is posting ready status" string?) In this case, if you have direct Secure Shell (SSH) access to the node, check the recent events to understand the error. Look within the `/var/log/messages` file. Or, generate the kubelet and container daemon log files by running the following shell commands:

Bash

```
# To check messages file,  
cat /var/log/messages  
  
# To check kubelet and containerd daemon logs,  
journalctl -u kubelet > kubelet.log  
journalctl -u containerd > containerd.log
```

After you run these commands, examine the messages and daemon log files for more information about the error.

Solution

Step 1: Check for changes in network-level

If all cluster nodes regressed to a **Not Ready** status, check whether any changes occurred at the network level. Examples of network-level changes include:

- Domain name system (DNS) changes
- Firewall rule changes, such as port, fully qualified domain names (FQDNs), and so on.
- Added network security groups (NSGs)
- Applied or changed route table configurations for AKS traffic

If there were changes at the network level, make any necessary corrections. If you have direct Secure Shell (SSH) access to the node, you can use the `curl` or `telnet` command to check the connectivity to [AKS outbound requirements](#). After you've fixed the issues, stop and restart the nodes. If the nodes stay in a healthy state after these fixes, you can safely skip the remaining steps.

Step 2: Stop and restart the nodes

If only a few nodes regressed to a **Not Ready** status, simply stop and restart the nodes. This action alone might return the nodes to a healthy state. Then, check [Azure](#)

[Kubernetes Service diagnostics overview](#) to determine whether there are any issues, such as the following issues:

- Node faults
- Source network address translation (SNAT) failures
- Node input/output operations per second (IOPS) performance issues
- Other issues

If the diagnostics don't discover any underlying issues and the nodes returned to Ready status, you can safely skip the remaining steps.

Step 3: Fix SNAT issues for public AKS API clusters

Did AKS diagnostics uncover any SNAT issues? If so, take some of the following actions, as appropriate:

- Check whether your connections remain idle for a long time and rely on the default idle time-out to release its port. If the connections exhibit this behavior, you might have to reduce the default time-out of 30 minutes.
- Determine how your application creates outbound connectivity. For example, does it use code review or packet capture?
- Determine whether this activity represents the expected behavior or, instead, it shows that the application is misbehaving. Use metrics and logs in Azure Monitor to substantiate your findings. For example, you can use the **Failed** category as a SNAT Connections metric.
- Evaluate whether appropriate patterns are followed.
- Evaluate whether you should mitigate SNAT port exhaustion by using extra outbound IP addresses and more allocated outbound ports. For more information, see [Scale the number of managed outbound public IPs](#) and [Configure the allocated outbound ports](#).

For more information about how to troubleshoot SNAT port exhaustion, see [Troubleshoot SNAT port exhaustion on AKS nodes](#).

Step 4: Fix IOPS performance issues

If AKS diagnostics uncover issues that reduce IOPS performance, take some of the following actions, as appropriate:

- To increase IOPS on virtual machine (VM) scale sets, choose a larger disk size that offers better IOPS performance by deploying a new node pool. Direct resizing VMSS directly isn't supported. For more information on resizing node pools, see [Resize node pools in Azure Kubernetes Service \(AKS\)](#).
- Increase the node SKU size for more memory and CPU processing capability.
- Consider using [Ephemeral OS](#).
- Limit the CPU and memory usage for pods. These limits help prevent node CPU consumption and out-of-memory situations.
- Use scheduling topology methods to add more nodes and distribute the load among the nodes. For more information, see [Pod topology spread constraints](#).

Step 5: Fix threading issues

Kubernetes components such as kubelets and [containerd runtimes](#) rely heavily on threading, and they spawn new threads regularly. If the allocation of new threads is unsuccessful, this failure can affect service readiness, as follows:

- The node status changes to **Not Ready**, but it's restarted by a remediator, and is able to recover.
- In the `/var/log/messages` and `/var/log/syslog` log files, there are repeated occurrences of the following error entries:

`| pthread_create failed: Resource temporarily unavailable by various processes`

The processes that are cited include containerd and possibly kubelet.

- The node status changes to **Not Ready** soon after the `pthread_create` failure entries are written to the log files.

Process IDs (PIDs) represent threads. The default number of PIDs that a pod can use might be dependent on the operating system. However, the default number is at least 32,768. This amount is more than enough PIDs for most situations. Are there any known application requirements for higher PID resources? If there aren't, then even an eight-fold increase to 262,144 PIDs might not be enough to accommodate a high-resource application.

Instead, identify the offending application, and then take the appropriate action. Consider other options, such as increasing the VM size or upgrading AKS. These actions

can mitigate the issue temporarily, but they aren't a guarantee that the issue won't reappear again.

To monitor the thread count for each control group (cgroup) and print the top eight cgroups, run the following shell command:

Bash

```
watch 'ps -e -w -o "thcount,cgname" --no-headers | awk "{a[\$2] += \$1} END{for (i in a) print a[i], i}" | sort --numeric-sort --reverse | head --lines=8'
```

For more information, see [Process ID limits and reservations](#).

Kubernetes offers two methods to manage PID exhaustion at the node level:

1. Configure the maximum number of PIDs that are allowed on a pod within a kubelet by using the `--pod-max-pids` parameter. This configuration sets the `pids.max` setting within the cgroup of each pod. You can also use the `--system-reserved` and `--kube-reserved` parameters to configure the system and kubelet limits, respectively.
2. Configure PID-based eviction.

 **Note**

By default, neither of these methods are set up. Additionally, you can't currently configure either method by using [Node configuration for AKS node pools](#).

Step 6: Use a higher service tier

You can make sure that the AKS API server has high availability by using a higher service tier. For more information, see the [Azure Kubernetes Service \(AKS\) Uptime SLA](#).

More information

- To view the health and performance of the AKS API server and kubelets, see [Managed AKS components](#).
- For general troubleshooting steps, see [Basic troubleshooting of node not ready failures](#).

Feedback

Was this page helpful?

 Yes

 No

Troubleshoot Node Not Ready failures that are followed by recoveries

Article • 03/19/2025

This article provides a guide to troubleshoot and resolve "Node Not Ready" issues in Azure Kubernetes Service (AKS) clusters. When a node enters a "NotReady" state, it can disrupt the application's functionality and cause it to stop responding. Typically, the node recovers automatically after a short period. However, to prevent recurring issues and maintain a stable environment, it's important to understand the underlying causes to be able to implement effective resolutions.

Cause

There are several scenarios that could cause a "NotReady" state to occur:

- The unavailability of the API server. This causes the readiness probe to fail. This prevents the pod from being attached to the service so that traffic is no longer forwarded to the pod instance.
- Virtual machine (VM) host faults. To determine whether VM host faults occurred, check the following information sources:
 - [AKS diagnostics](#)
 - [Azure status ↗](#)
 - Azure notifications (for any recent outages or maintenance periods)

Resolution

To resolve this issue, follow these steps:

1. Run `kubectl describe node <node-name>` to review detail information about the node's status. Look for any error messages or warnings that might indicate the root cause of the issue.
2. Check the API server availability by running the `kubectl get apiservices` command. Make sure that the readiness probe is correctly configured in the deployment YAML file.
3. Verify the node's network configuration to make sure that there are no connectivity issues.
4. Check the node's resource usage, such as CPU, memory, and disk, to identify potential constraints. For more informations see [Monitor your Kubernetes cluster](#)

performance with Container insights

For further steps, see [Basic troubleshooting of Node Not Ready failures](#).

Prevention

To prevent this issue from occurring in the future, take one or more of the following actions:

- Make sure that your service tier is fully paid for.
- Reduce the number of `watch` and `get` requests to the API server.

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#)

Troubleshoot common node auto-repair errors

Article • 10/25/2024

This article provides potential causes and solutions to common node auto-repair errors, and outlines best practices for monitoring the node auto-repair process.

Overview

When Azure Kubernetes Service (AKS) detects a node with a `NotReady` status for more than five minutes, it attempts to automatically repair the node. Node auto-repair is a best-effort service. It doesn't guarantee that the node can be restored to a healthy state. For more information, see [node auto-repair process](#).

During the node auto-repair process, AKS initiates `reboot`, `reimage`, and `redeploy` actions on your unhealthy node. Errors can occur due to various reasons and error codes are discovered through [Kubernetes events](#). You can use Kubernetes events to monitor the status of your node and the auto-repair actions.

Kubernetes events

To identify the type of a node auto-repair error, check the following Kubernetes events:

[] Expand table

Reason	Event message	Description
NodeRebootError	Node auto-repair reboot action failed due to an operation failure: [error code here]	Emitted when there's an error with the <code>reboot</code> action.
NodeReimageError	Node auto-repair reimage action failed due to an operation failure: [error code here]	Emitted when there's an error with the <code>reimage</code> action.
NodeRedeployError	Node auto-repair redeploy action failed due to an operation failure: [error code here]	Emitted when there's an error with the <code>redeploy</code> action.

! Note

Because your node is already in an unhealthy state prior to the auto-repair process, in most cases, node auto-repair errors don't impact your cluster or applications. When you encounter node auto-repair errors, we recommend that you try to repair the node by following the instructions in [Basic troubleshooting of Node Not Ready failures](#). If you can't restore it to a `Succeeded` status and see persistent errors reported by node auto-repair, contact [Azure support](#) for assistance.

Common error codes

[] [Expand table](#)

Error code	Cause and solution
VMExtensionProvisioningError	One or more virtual machine (VM) extensions failed to be provisioned on the VM. For more information about possible error types and troubleshooting steps, see Troubleshoot the ERR_VHD_FILE_NOT_FOUND error code (124) . To determine the exact VM extension provisioning error on your node, view error details in the Azure portal .
InvalidParameter	This error occurs if the node auto-repair process tries to access a node that no longer exists.
scaleSetNameAndInstanceIDFromProviderID failed	This issue occurs when the node isn't provisioned correctly.
ManagedIdentityCredential authentication failed	This issue occurs when the node isn't initialized correctly.
VMRedeploymentFailed	This error occurs when you try to redeploy the node. In this case, your node pool might enter a failed state. For more information about potential causes and troubleshooting steps, see Troubleshoot Azure Kubernetes Service clusters or nodes in a failed state .
TooManyVMRedeploymentRequests	This error occurs when your cluster exceeds the limit for VM redeployment requests. <code>Redeploy</code> is one of the node auto-repair actions. This error means that the <code>redeploy</code> action can't repair your node. To troubleshoot the Node Not Ready issue, see Basic troubleshooting of Node Not Ready failures .

Error code	Cause and solution
OutboundConnectivityNotEnabledOnVMSS	<p>This error occurs when your node or overall Virtual Machine Scale Set doesn't have outbound access enabled. To resolve this issue, enable secure outbound access for your scale set by using a method that's best suited for your application. For more information, see "OutboundConnectivityNotEnabledOnVM. No outbound connectivity configured for virtual machine."</p>

Best practices for monitoring node auto-repair

- AKS stores Kubernetes events from the past hour by default. We recommend enabling [Container Insights](#) so that you can store events for up to 90 days. You can also [query events and configure alerts](#) to quickly detect node auto-repair errors.
- Node auto-repair is a best-effort service. It doesn't guarantee that your node can be restored to a `Ready` status. We recommend that you actively monitor on and set alerts for Node Not Ready issues, and troubleshoot and resolve these issues yourself. For more information, see [basic troubleshooting of Node Not Ready issues](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

Troubleshoot connection issues to an app that's hosted in an AKS cluster

Article • 10/08/2024

In current dynamic cloud environments, ensuring seamless connectivity to applications hosted in Azure Kubernetes Service (AKS) clusters is crucial for maintaining optimal performance and user experience. This article covers how to troubleshoot and resolve connectivity issues caused by various factors, including application-side problems, network policies, Network security group (NSG) rules or others.

ⓘ Note

To troubleshoot common issues when you try to connect to the AKS API server, see [Basic troubleshooting of cluster connection issues with the API server](#).

Prerequisites

- The Client URL ([cURL](#)) tool, or a similar command-line tool.
- The [apt-get](#) command-line tool for handling packages.
- The [Netcat](#) (`nc`) command-line tool for TCP connections.
- The Kubernetes [kubectl](#) tool, or a similar tool to connect to the cluster. To install kubectl by using [Azure CLI](#), run the `az aks install-cli` command.

Factors to consider

This section covers troubleshooting steps to take if you're having issues when you try to connect to the application that's hosted in an AKS cluster.

In any networking scenario, administrators should consider the following important factors when troubleshooting:

- What's the source and the destination for a request?
- What are the hops in between the source and the destination?
- What's the request-response flow?

- Which hops have extra security layers on top, such as the following items:
 - Firewall
 - Network security group (NSG)
 - Network policy

When you check each component, [get and analyze HTTP response codes](#). These codes are useful to identify the nature of the issue, and are especially helpful in scenarios in which the application responds to HTTP requests.

If other troubleshooting steps don't provide any conclusive outcome, take packet captures from the client and server. Packet captures are also useful when non-HTTP traffic is involved between the client and server. For more information about how to collect packet captures for AKS environment, see the following articles in the data collection guide:

- [Capture a TCP dump from a Linux node in an AKS cluster](#).
- [Capture a TCP dump from a Windows node in an AKS cluster](#).
- [Capture TCP packets from a pod on an AKS cluster](#).

Knowing how to get the HTTP response codes and take packet captures makes it easier to troubleshoot a network connectivity issue.

Basic network flow for applications on AKS

In general, when applications are exposed using the Azure Load Balancer service type, the request flow to access them is as follows:

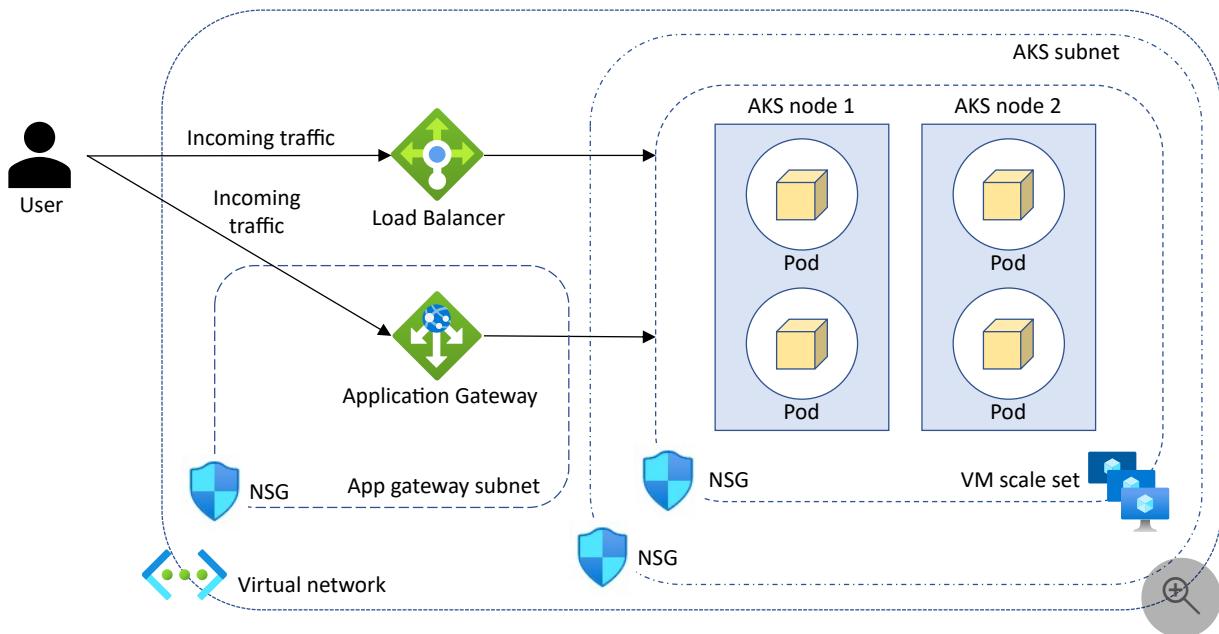
Client >> DNS name >> AKS load balancer IP address >> AKS nodes >> Pods

There are other possible situations in which extra components might be involved. For example:

- The managed NGINX ingress with the [application routing add-on](#) feature is enabled.
- The application gateway is used through the [Application Gateway Ingress Controller](#) (AGIC) instead of Azure Load Balancer.
- Azure Front Door and API Management might be used on top of the load balancer.
- The process uses an internal load balancer.
- The connection might not end at the pod and the requested URL. This could depend on whether the pod can connect to another entity, such as a database or any other service in the same cluster.

It's important to understand the request flow for the application.

A basic request flow to applications in an AKS cluster would resemble the flow that's shown in the following diagram.



Inside-out troubleshooting

Troubleshooting connectivity issues might involve many checks, but the *inside-out* approach can help find the source of the issue and identify the bottleneck. In this approach, you start at the pod itself, checking whether the application is responding on the pod's IP address. Then, check each component in turn up to the end client.

Step 1: Check whether the pod is running and the app or container inside the pod is responding correctly

To determine whether the pod is running, run one of the following [kubectl get](#) commands:

Bash

```
# List pods in the specified namespace.  
kubectl get pods -n <namespace-name>  
  
# List pods in all namespaces.  
kubectl get pods -A
```

What if the pod isn't running? In this case, check the pod events by using the [kubectl describe](#) command:

```
Bash
```

```
kubectl describe pod <pod-name> -n <namespace-name>
```

If the pod isn't in a `Ready` or `Running` state, or it restarted many times, check the `kubectl describe` output. The events will reveal any issues that prevent you from being able to start the pod. Or, if the pod has started, the application inside the pod might have failed, causing the pod to be restarted. [Troubleshoot the pod accordingly](#) to make sure that it's in a suitable state.

If the pod is running, it can also be useful to check the logs of the containers that are inside the pod. Run the following series of [kubectl logs](#) commands:

```
Bash
```

```
kubectl logs <pod-name> -n <namespace-name>

# Check logs for an individual container in a multicontainer pod.
kubectl logs <pod-name> -n <namespace-name> -c <container-name>

# Dump pod logs (stdout) for a previous container instance.
kubectl logs <pod-name> --previous

# Dump pod container logs (stdout, multicontainer case) for a previous
# container instance.
kubectl logs <pod-name> -c <container-name> --previous
```

Is the pod running? In this case, test the connectivity by starting a test pod in the cluster. From the test pod, you can directly access the application's pod IP address and check whether the application is responding correctly. Run the `kubectl run`, `apt-get`, and `cURL` commands as follows:

```
Bash
```

```
# Start a test pod in the cluster:
kubectl run -it --rm aks-ssh --image=debian:stable

# After the test pod is running, you will gain access to the pod.
# Then you can run the following commands:
apt-get update -y && apt-get install dnsutils -y && apt-get install curl -y
&& apt-get install netcat-traditional -y

# After the packages are installed, test the connectivity to the application
```

```
pod:  
curl -Iv http://<pod-ip-address>:<port>
```

For applications that listen on other protocols, you can install relevant tools inside the test pod like the netcat tool, and then check the connectivity to the application pod by running the following command:

Bash

```
# After the packages are installed, test the connectivity to the application  
pod using netcat/nc command:  
nc -z -v <pod-ip-address> <port>
```

For more commands to troubleshoot pods, see [Debug running pods](#).

Step 2: Check whether the application is reachable from the service

For scenarios in which the application inside the pod is running, you can focus mainly on troubleshooting how the pod is exposed.

Is the pod exposed as a service? In this case, check the service events. Also, check whether the pod IP address and application port are available as an endpoint in the service description:

Bash

```
# Check the service details.  
kubectl get svc -n <namespace-name>  
  
# Describe the service.  
kubectl describe svc <service-name> -n <namespace-name>
```

Check whether the pod's IP address is present as an endpoint in the service, as in the following example:

Console

```
$ kubectl get pods -o wide # Check the pod's IP address.  
NAME        READY   STATUS    RESTARTS   AGE     IP          NODE  
my-pod      1/1     Running   0          12m    10.244.0.15   aks-  
agentpool-000000-vmss000000  
  
$ kubectl describe service my-cluster-ip-service # Check the endpoints in  
the service.  
Name:           my-cluster-ip-service
```

```
Namespace:           default
Selector:            app=my-pod
Type:                ClusterIP
IP Family Policy:   SingleStack
IP Families:        IPv4
IP:                 10.0.174.133
IPs:                10.0.174.133
Port:               <unset>  80/TCP
TargetPort:          80/TCP
Endpoints:          10.244.0.15:80    # <-- Here
```

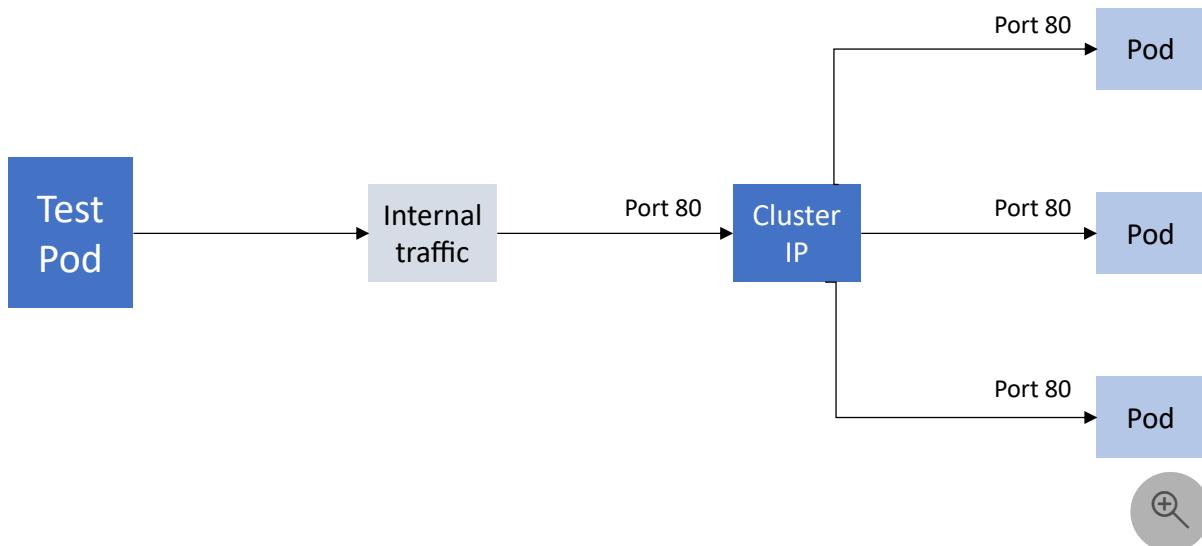
```
$ kubectl get endpoints # Check the endpoints directly for verification.
NAME                  ENDPOINTS          AGE
my-cluster-ip-service 10.244.0.15:80    14m
```

If the endpoints aren't pointing to the correct pod IP address, verify the `Labels` and `Selectors` for the pod and the service.

Are the endpoints in the service correct? If so, access the service, and check whether the application is reachable.

Access the ClusterIP service

For the `ClusterIP` service, you can start a test pod in the cluster and access the service IP address:



Bash

```
# Start a test pod in the cluster:
kubectl run -it --rm aks-ssh --image=debian:stable
```

```
# After the test pod is running, you will gain access to the pod.  
# Then, you can run the following commands:  
apt-get update -y && apt-get install dnsutils -y && apt-get install curl -y  
&& apt-get install netcat-traditional -y  
  
# After the packages are installed, test the connectivity to the service:  
curl -Iv http://<service-ip-address>:<port>
```

For applications that listen on other protocols, you can install relevant tools inside the test pod like the netcat tool, and then check the connectivity to the application pod by running the following command:

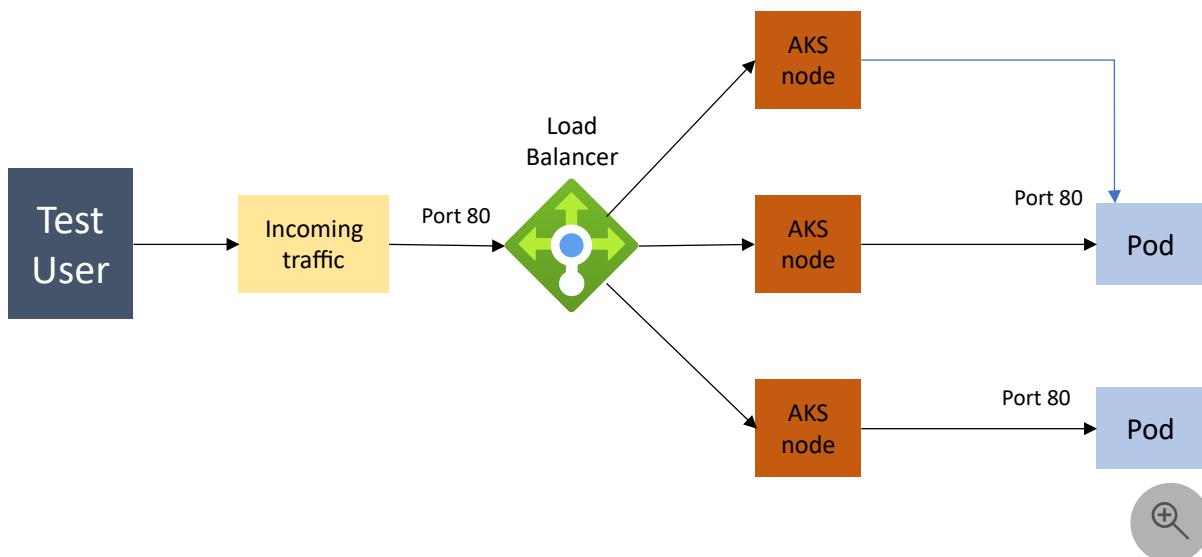
Bash

```
# After the packages are installed, test the connectivity to the application  
pod using netcat/nc command:  
nc -z -v <pod-ip-address> <port>
```

If the previous command doesn't return an appropriate response, check the service events for any errors.

Access the LoadBalancer service

For the `LoadBalancer` service, you can access the load balancer IP address from outside the cluster.



Bash

```
curl -Iv http://<service-ip-address>:<port>
```

For applications that listen on other protocols, you can install relevant tools inside the test pod like the netcat tool, and then check the connectivity to the application pod by running the following command:

Bash

```
nc -z -v <pod-ip-address> <port>
```

Does the `LoadBalancer` service IP address return a correct response? If it doesn't, follow these steps:

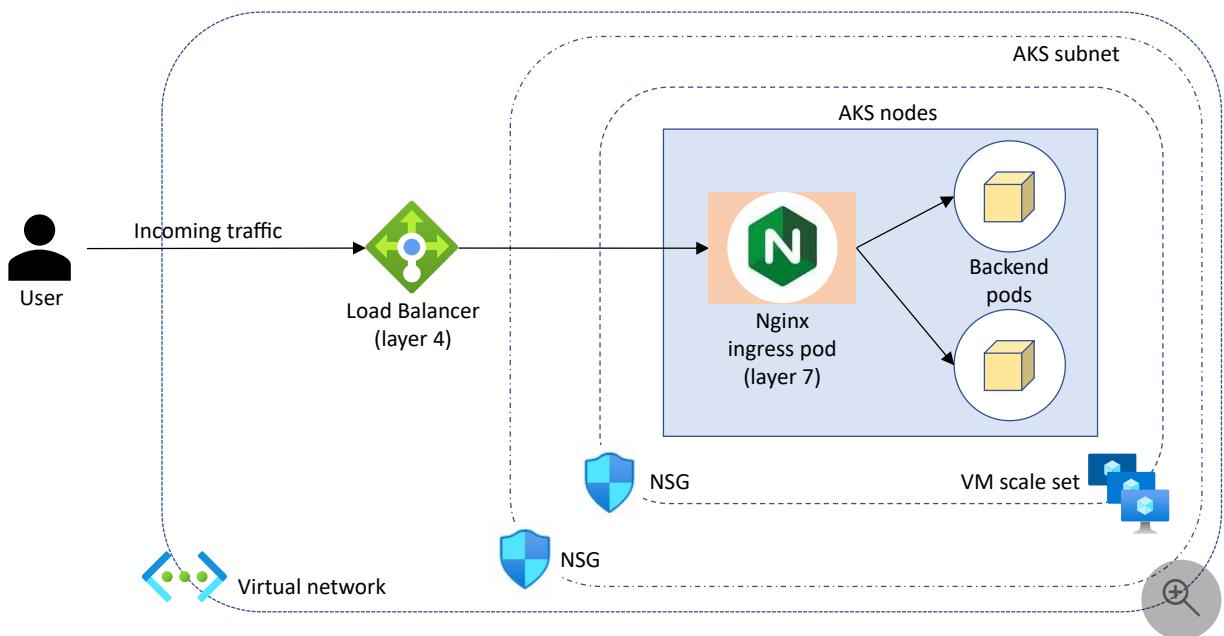
1. Verify the events of the service.
2. Verify that the network security groups (NSGs) that are associated with the AKS nodes and AKS subnet allow the incoming traffic on the service port.

For more commands to troubleshoot services, see [Debug services ↗](#).

Scenarios that use an ingress instead of a service

For scenarios in which the application is exposed by using an `Ingress` resource, the traffic flow resembles the following progression:

Client >> DNS name >> Load balancer or application gateway IP address >>
Ingress controller pods inside the cluster >> Service or pods



You can apply the inside-out approach of troubleshooting here, too. You can also check the ingress kubernetes resource and ingress controller details for more information:

```
Console

$ kubectl get ing -n <namespace-of-ingress> # Checking the ingress details and events.
NAME          CLASS      HOSTS           ADDRESS
PORTS        AGE
hello-world-ingress   <none>    myapp.com       20.84.x.x   80,
443         7d22h

$ kubectl describe ing -n <namespace-of-ingress> hello-world-ingress
Name:            hello-world-ingress
Namespace:       <namespace-of-ingress>
Address:         20.84.x.x
Default backend: default-http-backend:80 (<error: endpoints "default-http-backend" not found>)
TLS:
  tls-secret terminates myapp.com
Rules:
  Host          Path  Backends
  ----          ----  -----
  myapp.com     /blog  blog-service:80 (10.244.0.35:80)
                 /store  store-service:80 (10.244.0.33:80)

Annotations:      cert-manager.io/cluster-issuer: letsencrypt
                  kubernetes.io/ingress.class: nginx
                  nginx.ingress.kubernetes.io/rewrite-target: /$1
                  nginx.ingress.kubernetes.io/use-regex: true

Events:
  Type  Reason  Age   From           Message
  ----  -----  ---   ----

```

Normal	Sync	5m41s	nginx-ingress-controller	Scheduled for sync
Normal	Sync	5m41s	nginx-ingress-controller	Scheduled for sync

This example contains an `Ingress` resource that:

- Listens on the `myapp.com` host.
- Has two `Path` strings configured.
- Routes to two `Services` in the back end.

Verify that the back-end services are running, and respond to the port that's mentioned in the ingress description:

Console

```
$ kubectl get svc -n <namespace-of-ingress>
NAME          TYPE        CLUSTER-IP      EXTERNAL-IP    PORT(S)
ingress-basic  ClusterIP   10.0.155.154  <none>        80/TCP
ingress-basic  ClusterIP   10.0.61.185   <none>        80/TCP
ingress-basic  LoadBalancer 10.0.122.148  20.84.x.x   80:30217/TCP,443:32464/TCP
```

Verify the logs for the ingress controller pods if there's an error:

Console

```
$ kubectl get pods -n <namespace-of-ingress> # Get the ingress controller
pods.
NAME          READY  STATUS
RESTARTS      AGE
aks-helloworld-one-56c7b8d79d-6zktl      1/1   Running   0
31h
aks-helloworld-two-58bbb47f58-rrcv7      1/1   Running   0
31h
nginx-ingress-ingress-nginx-controller-9d8d5c57d-9vn8q  1/1   Running   0
31h
nginx-ingress-ingress-nginx-controller-9d8d5c57d-grzdr  1/1   Running   0
31h

$ # Check logs from the pods.
$ kubectl logs -n ingress-basic nginx-ingress-ingress-nginx-controller-
9d8d5c57d-9vn8q
```

What if the client is making requests to the ingress host name or IP address, but no entries are seen in the logs of the ingress controller pod? In this case, the requests

might not be reaching the cluster, and the user might be receiving a [Connection Timed Out](#) error message.

Another possibility is that the components on top of the ingress pods, such as Load Balancer or Application Gateway, aren't routing the requests to the cluster correctly. If this is true, you can check the back-end configuration of these resources.

If you receive a [Connection Timed Out](#) error message, check the network security group that's associated with the AKS nodes. Also, check the AKS subnet. It could be blocking the traffic from the load balancer or application gateway to the AKS nodes.

For more information about how to troubleshoot ingress (such as Nginx Ingress), see [ingress-nginx troubleshooting](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Feedback

Was this page helpful?

 Yes

 No

A custom network security group blocks traffic

Article • 09/16/2024

When you access an application that's hosted on an Azure Kubernetes Service (AKS) cluster, you receive a "Timed out" error message. This error can occur even if the application is running and the rest of the configuration appears to be correct.

Prerequisites

- The Kubernetes [kubectl](#) tool, or a similar tool, to connect to the cluster. To install kubectl by using [Azure CLI](#), run the `az aks install-cli` command.
- The Client URL ([cURL](#)) tool, or a similar command-line tool.
- The [apt-get](#) command-line tool for handling packages.

Symptoms

If you run the following [kubectl get](#) and cURL commands, you experience "Timed out" errors that resemble the following console output:

```
Console

$ kubectl get pods
NAME                  READY   STATUS    RESTARTS   AGE
my-deployment-66648877fc-v78jm   1/1     Running   0          5m53s

$ kubectl get service
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)
AGE
my-loadbalancer-service   LoadBalancer   10.0.107.79   10.81.x.x   80:31048/TCP
4m14s

$ curl -Iv http://10.81.124.39 # Use an IP address that fits the "EXTERNAL-IP"
pattern.
*   Trying 10.81.x.x:80...
* connect to 10.81.x.x port 80 failed: Timed out
* Failed to connect to 10.81.x.x port 80 after 21033 ms: Timed out
* Closing connection 0
curl: (28) Failed to connect to 10.81.x.x port 80 after 21033 ms: Timed out
```

Cause

If you experience the same "Timed out" error every time, that usually suggests that a network component is blocking the traffic.

To troubleshoot this issue, you can start by checking access to the pod, and then move on to the client in an *inside-out* approach.

To check the pod, run the following `kubectl get` and `kubectl describe` commands:

```
Console

$ kubectl get pods -o wide
NAME                               READY   STATUS    RESTARTS   AGE     IP
NODE
my-deployment-66648877fc-v78jm   1/1     Running   0          53s    172.25.0.93
aks-agentpool-42617579-vmss000000

$ kubectl describe pod my-deployment-66648877fc-v78jm # Specify the pod name from
the previous command.
...
...
Events:
  Type  Reason  Age   From            Message
  ----  -----  --   --   -----
  Normal Scheduled  117s  default-scheduler  Successfully assigned default/my-
deployment-66648877fc-v78jm to aks-agentpool-42617579-vmss000000
  Normal Pulling   116s  kubelet         Pulling image "httpd"
  Normal Pulled    116s  kubelet         Successfully pulled image "httpd" in
183.532816ms
  Normal Created   116s  kubelet         Created container webserver
  Normal Started   116s  kubelet         Started container webserver
```

Based on this output, the pod seems to be running correctly, without any restarts.

Open a test pod to check access to the application pod. Run the following `kubectl get`, `kubectl run`, `apt-get`, and cURL commands:

```
Console

$ kubectl get pods -o wide # Get the pod IP address.
NAME                               READY   STATUS    RESTARTS   AGE     IP
NODE
my-deployment-66648877fc-v78jm   1/1     Running   0          7m45s  172.25.0.93
aks-agentpool-42617579-vmss000000

$ kubectl run -it --rm aks-ssh --image=debian:stable # Launch the test pod.
If you don't see a command prompt, try pressing enter.
$ root@aks-ssh:

$ # Install packages inside the test pod.
$ root@aks-ssh: apt-get update -y && apt-get install dnsutils -y && apt-get install
curl -y
Get:1 http://deb.debian.org/debian bullseye InRelease [116 kB]
Get:2 http://deb.debian.org/debian bullseye-updates InRelease [39.4 kB]
...
...
Running hooks in /etc/ca-certificates/update.d...
done.

$ # Try to check access to the pod using the pod IP address from the "kubectl get"
output.
```

```
$ curl -Iv http://172.25.0.93
*   Trying 172.25.0.93:80...
* Connected to 172.25.0.93 (172.25.0.93) port 80 (#0)
...
...
< HTTP/1.1 200 OK
HTTP/1.1 200 OK
...
...
* Connection #0 to host 172.25.0.93 left intact
```

The pod is accessible directly. Therefore, the application is running.

The defined service is a `LoadBalancer` type. This means that the request flow from the end client to the pod will be as follows:

Client >> Load balancer >> AKS node >> Application pod

In this request flow, we can block the traffic through the following components:

- Network policies in the cluster
- The network security group (NSG) for the AKS subnet and AKS node

To check network policy, run the following `kubectl get` command:

Console

```
$ kubectl get networkpolicy --all-namespaces
NAMESPACE      NAME          POD-SELECTOR      AGE
kube-system    konnectivity-agent  app=konnectivity-agent  3h8m
```

Only the AKS default policy exists. Therefore, network policy doesn't seem to be blocking the traffic.

To check the NSGs and their associated rules by using AKS, follow these steps:

1. In the [Azure portal](#), search for and select **Virtual machine scale sets**.
2. In the list of scale set instances, select the one that you're using.
3. In the menu pane of your scale set instance, select `Networking`.

The **Networking** page for the scale set instance appears. In the **Inbound port rules** tab, two sets of rules are displayed that are based on the two NSGs that act on the scale set instance:

- The first set is composed of NSG rules at the subnet level. These rules are displayed under the following note heading:

Network security group <*my-aks-nsg*> (attached to subnet: <*my-aks-subnet*>)

This arrangement is common if a custom virtual network and custom subnet for the AKS cluster are used. The set of rules at the subnet level might resemble the following table.

[+] Expand table

Priority	Name	Port	Protocol	Source	Destination	Action
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

- The second set is composed of NSG rules at the network adapter level. These rules are displayed under the following note heading:

Network security group *aks-agentpool-<agentpool-number>-nsg* (attached to network interface: *aks-agentpool-<vm-scale-set-number>-vmss*)

This NSG is applied by the AKS cluster, and it's managed by AKS. The corresponding set of rules might resemble the following table.

[+] Expand table

Priority	Name	Port	Protocol	Source	Destination	Action
500	<guid>-TCP-80-Internet	80	TCP	Internet	10.81.x.x	Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

At the network adapter level, there's an NSG inbound rule for TCP at IP address 10.81.x.x on port 80 (highlighted in the table). However, an equivalent rule is missing from the rules for the NSG at the subnet level.

Why didn't AKS apply the rule to the custom NSG? Because AKS doesn't apply NSGs to its subnet, and it won't modify any of the NSGs that are associated with that subnet. AKS will modify the NSGs only at the network adapter level. For more information, see [Can I configure NSGs with AKS?](#).

Solution

If the application is enabled for access on a certain port, you must make sure that the custom NSG allows that port as an **Inbound** rule. After the appropriate rule is added in the custom NSG at the subnet level, the application is accessible.

Console

```
$ curl -Iv http://10.81.x.x
*   Trying 10.81.x.x:80...
* Connected to 10.81.x.x (10.81.x.x) port 80 (#0)
...
...
< HTTP/1.1 200 OK
HTTP/1.1 200 OK
...
...
* Connection #0 to host 10.81.x.x left intact
```

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Resolve "az aks command invoke" failures

Article • 05/13/2025

This article describes how to resolve [az aks command invoke](#) failures in Microsoft Azure CLI so that you can successfully connect to any Azure Kubernetes Service (AKS) cluster, especially to a [private AKS cluster](#).

Other connection methods need to use extra configuration components, as shown in the following table.

[+] [Expand table](#)

Connection methods	Extra configuration component
Virtual network	Virtual private network (VPN)
Peered network	Azure ExpressRoute
Private endpoint	Jumpbox

The `az aks command invoke` Azure CLI command is an alternative way of connecting to a cluster that doesn't require extra configuration components.

When you run the `az aks command invoke` command, Azure CLI automatically creates a `command-<ID>` pod in the `aks-command` namespace to access the AKS cluster and retrieve the required information.

Prerequisites

- [Azure CLI](#).
- The Kubernetes [kubectl](#) tool. To install kubectl by using Azure CLI, run the [az aks install-cli](#) command.

Symptoms

The following table lists common `az aks command invoke` error messages. Each error message has a link to the section that describes why the error is occurring, and how to fix it.

[+] [Expand table](#)

Error message	Link
Operation returned an invalid status 'Not Found'	Cause 1: The pod can't be created because of node or resource constraints
Failed to run command in managed cluster due to kubernetes failure. details: admission webhook "validation.gatekeeper.sh" denied the request: <policy-specific-message>	Cause 2: Azure Policy doesn't allow the pod creation
Error from server (Forbidden): namespaces is forbidden: User "<ID>" cannot list resource "<resource>" in API group "" at the cluster scope	Cause 3: Required roles aren't granted
Failed to connect to MSI. Please make sure MSI is configured correctly.	Cause 4: There's a Cloud Shell issue
Get Token request returned: Response [400];	

Cause 1: The pod can't be created because of node or resource constraints

The operation returns a `Not Found` status because the `command-<ID>` pod can't reach a successful state, such as `Running`. (In many cases, the pod stays in the `Pending` state.) In this case, the nodes aren't able to schedule the pod. This scenario can have different causes, such as the following causes:

- Resource constraints
- Nodes that have a `NotReady` or `SchedulingDisabled` state
- Nodes that have taints that the pod can't tolerate
- Other causes

Here are sample error messages for `KubernetesPerformanceError` or `KubernetesOperationError`:

Output
<pre>(KubernetesPerformanceError) Failed to run command due to cluster perf issue, container command-357ebsdfsd342869 in aks-command namespace did not start within 30s on your cluster, retry may helps. If issue persist, you may need to tune your cluster with better performance (larger node/paid tier). Code: KubernetesPerformanceError Message: Failed to run command due to cluster perf issue, container command- 357ebc50d40c47a4a247ab6e067d2869 in aks-command namespace did not start within 30s on your cluster, retry may helps. If issue persist, you may need to tune your cluster with better performance (larger node/paid tier).</pre>

Solution 1: Change the configuration so that you can schedule and run the pod

Make sure that the `command-<ID>` pod can be scheduled and run by adjusting the configuration. For example:

- Increase the node pool size and make sure it has no pod scheduling constraints like taints so that the `command-<ID>` pod can be deployed.
- Adjust resource requests and limits in your pod specifications.

Cause 2: Azure Policy doesn't allow the pod creation

If you have specific Azure policies, the `az aks command invoke` command can fail because of a disallowed configuration in the `command-<ID>` pod. For example, you might have an Azure policy that requires a read-only root file system or other specific configuration.

Solution 2: Exempt the namespace for policies that prohibit pod creation

We recommend that you exempt the `aks-command` namespace for the associated Azure policies that don't allow the pod creation. For more information about exemption, see [Understand scope in Azure Policy](#)

To exempt an Azure Policy:

1. In the [Azure portal](#), search for and select **Policy**.
2. In the **Policy** navigation pane, locate the **Authoring** section, and then select **Assignments**.
3. In the table of assignments, find the row that contains the **Assignment name** that you want to change, and then select the name of the assignment.
4. In the policy assignment page for that assignment, select **Edit assignment**.
5. Select the **Parameters** tab.
6. Clear the **Only show parameters that need input or review** option.
7. In the **Namespace exclusions** box, add the `aks-command` namespace to the list of namespaces to be excluded.

Alternatively, if the policy isn't a built-in policy, you can check the configuration of the `command-<ID>` pod, and adjust the policy as necessary. To explore the pod's YAML configuration, run the

following command:

```
Bash
```

```
kubectl get pods command-<ID> --namespace aks-command --output yaml
```

You can exempt the `aks-command` namespace from restrictive policies by running the following command:

```
Bash
```

```
az policy exemption create --name ExemptAksCommand --scope  
/subscriptions/{subscription-id}/resourceGroups/{resource-  
group}/providers/Microsoft.ContainerService/managedClusters/{aks-cluster} --  
policyAssignment /subscriptions/{subscription-  
id}/providers/Microsoft.Authorization/policyAssignments/{policy-assignment-id}
```

Cause 3: Required roles aren't granted

To use the `az aks command invoke` command, you must have access to the following roles on the cluster:

- `Microsoft.ContainerService/managedClusters/runCommand/action`
- `Microsoft.ContainerService/managedClusters/commandResults/read`

If you don't have these roles, the `az aks command invoke` command can't retrieve the required information.

Solution 3: Add the required roles

To resolve this issue, follow these steps:

1. Add the `Microsoft.ContainerService/managedClusters/runCommand/action` and `Microsoft.ContainerService/managedClusters/commandResults/read` roles.

2. Assign the necessary roles to the user:

```
Bash
```

```
az role assignment create --assignee {user-principal-name} --role "Azure  
Kubernetes Service Cluster User Role" --scope /subscriptions/{subscription-  
id}/resourceGroups/{resource-  
group}/providers/Microsoft.ContainerService/managedClusters/{aks-cluster}
```

Cause 4: There's a Cloud Shell issue

The `az aks command invoke` command isn't processed as expected when it's run directly in the [Azure Cloud Shell](#) environment. This is a known issue in Cloud Shell.

Solution 4a: Run the az login command first

In Cloud Shell, run the `az login` command before you run the `az aks command invoke` command. For example:

Bash

```
az login
az aks command invoke --resource-group {resource-group} --name {aks-cluster} --
  command "kubectl get pods"
```

Solution 4b: Run the command on a local computer or a virtual machine

Run the `az aks command invoke` command on a local computer or any virtual machine (VM) that has Azure CLI installed.

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Get and analyze HTTP response codes

Article • 09/27/2024

If an application responds to HTTP or HTTPS requests, you can check the HTTP response codes to determine the behavior of the application.

Prerequisites

- The Client URL ([cURL](#)) tool, or another similar command-line tool.
- The Kubernetes [kubectl](#) tool, or a similar tool to connect to the cluster. To install kubectl by using [Azure CLI](#), run the `az aks install-cli` command.

Get HTTP response codes by using cURL

The [cURL](#) command-line tool can send an HTTP request to an application endpoint and get the response. For a load balancer service (that responds on the path "/" on port 80), a curl request can be initiated by running the following command:

Bash

```
curl -Iv http://<load-balancer-service-ip-address>:80/
```

For example, you can use cURL together with the [kubectl get](#) command, as follows:

Console

```
$ kubectl get service
NAME           TYPE      CLUSTER-IP      EXTERNAL-IP
PORT(S)        AGE
my-loadbalancer-service   LoadBalancer  10.0.81.95    20.62.x.x
80:32131/TCP  18h

$ curl -Iv http://20.62.x.x:80/
* Trying 20.62.x.x:80...
* Connected to 20.62.x.x (20.62.x.x) port 80 (#0)
> HEAD / HTTP/1.1
> Host: 20.62.x.x
> User-Agent: curl/7.79.1
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
HTTP/1.1 200 OK
< ...
```

```
...
< Server: Apache/2.4.52 (Unix)
Server: Apache/2.4.52 (Unix)
< ...
...
<
* Connection #0 to host 20.62.x.x left intact
```

The response from this URI is `HTTP 200`.

Get HTTP response codes by using a browser

You can also get the HTTP response of an HTTP endpoint from a browser. Follow these steps:

1. In a browser window, press `Ctrl+Shift+I` or `F12`. The developer tools window or pane appears.
2. Select the **Network** tab, and then access the endpoint. The details about the HTTP response appear in the developer tools window or pane.

Get HTTP response codes by issuing API requests

To make API requests to the application and get details about the response, you can choose from many other command-line and GUI tools. These tools include the following:

[+] Expand table

Tool	Link
Postman	Postman API platform ↗
wget	GNU Wget 1.21.1-dirty Manual ↗
PowerShell	Invoke-WebRequest cmdlet

After you get an HTTP response code, start troubleshooting to better understand the application's behavior. For more information about the HTTP status codes and the behavior that they indicate, see the following content:

[+] Expand table

Information source	Link
Internet Assigned Numbers Authority (IANA)	Hypertext Transfer Protocol (HTTP) status code registry
Mozilla	HTTP response status codes
Wikipedia	List of HTTP status codes

The following HTTP status codes might indicate the listed issues.

[Expand table](#)

HTTP status code	Issue
4xx	An issue affects the client request. For example, the requested page doesn't exist, or the client doesn't have permission to access the page. OR A network blocker exists between the client and the server. For example, traffic is being blocked by a network security group or a firewall.
5xx	An issue affects the server. For example, the application is down, or a gateway isn't working.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Intermittent time-outs or server issues when accessing the application on AKS

Article • 09/19/2024

This article describes how to troubleshoot intermittent connectivity issues that affect your applications that are hosted on an Azure Kubernetes Service (AKS) cluster.

Prerequisites

- The Client URL ([cURL ↗](#)) tool, or a similar command-line tool.
- The Kubernetes [kubectl ↗](#) tool, or a similar tool to connect to the cluster. To install kubectl by using [Azure CLI](#), run the `az aks install-cli` command.

Symptoms

When you run a cURL command, you occasionally receive a "Timed out" error message. The output might resemble the following text:

```
Console

$ # One connection is successful, which results in a HTTP 200 response.
$ curl -Iv http://20.62.x.x
*   Trying 20.62.x.x:80...
* Connected to 20.62.x.x (20.62.x.x) port 80 (#0)
...
...
< HTTP/1.1 200 OK
HTTP/1.1 200 OK

$ # Another connection is unsuccessful, because it gets timed out.
$ curl -Iv http://20.62.x.x
*   Trying 20.62.x.x:80...
* connect to 20.62.x.x port 80 failed: Timed out
* Failed to connect to 20.62.x.x port 80 after 21050 ms: Timed out
* Closing connection 0
curl: (28) Failed to connect to 20.62.x.x port 80 after 21050 ms: Timed out

$ # Then the next connection is again successful.
$ curl -Iv http://20.62.x.x
*   Trying 20.62.x.x:80...
* Connected to 20.62.x.x (20.62.x.x) port 80 (#0)
...
...
```

```
< HTTP/1.1 200 OK  
HTTP/1.1 200 OK
```

Cause

Intermittent time-outs suggest component performance issues, as opposed to networking problems.

In this scenario, it's important to check the usage and health of the components. You can use the *inside-out* technique to check the status of the pods. Run the [kubectl top ↗](#) and [kubectl get ↗](#) commands, as follows:

Console

```
$ kubectl top pods # Check the health of the pods and the nodes.  
NAME                      CPU(cores)   MEMORY(bytes)  
my-deployment-fc94b7f98-m9z21    1m           32Mi  
  
$ kubectl top nodes  
NAME                      CPU(cores)   CPU%     MEMORY(bytes)  
MEMORY%  
aks-agentpool-42617579-vmss000000    120m        6%      2277Mi          49%  
  
$ kubectl get pods # Check the state of the pod.  
NAME          READY   STATUS    RESTARTS   AGE  
my-deployment-fc94b7f98-m9z21    2/2     Running   1          108s
```

The output shows that the current usage of the pods and nodes appears to be acceptable.

Although the pod is in the `Running` state, one restart occurs after the first 108 seconds of the pod running. This occurrence might indicate that some issues affect the pods or containers that run in the pod.

If the issue persists, the status of the pod changes after some time:

Console

```
$ kubectl get pods  
NAME          READY   STATUS    RESTARTS   AGE  
my-deployment-fc94b7f98-m9z21    1/2     CrashLoopBackOff   42          3h53m
```

This example shows that the `Ready` state is changed, and there are several restarts of the pod. One of the containers is in `CrashLoopBackOff` state.

This situation occurs because the container fails after starting, and then Kubernetes tries to restart the container to force it to start working. However, if the issue persists, the application continues to fail after it runs for some time. Kubernetes eventually changes the status to `CrashLoopBackOff`.

To check the logs for the pod, run the following [kubectl logs](#) commands:

Console

```
$ kubectl logs my-deployment-fc94b7f98-m9z21
error: a container name must be specified for pod my-deployment-fc94b7f98-
m9z21, choose one of: [webserver my-app]

$ # Since the pod has more than one container, the name of the container has
# to be specified.
$ kubectl logs my-deployment-fc94b7f98-m9z21 -c webserver
[...] [mpm_event:notice] [pid 1:tid 140342576676160] AH00489: Apache/2.4.52
(Unix) configured -- resuming normal operations
[...] [core:notice] [pid 1:tid 140342576676160] AH00094: Command line:
'httpd -D FOREGROUND'
10.244.0.1 - - ... "GET / HTTP/1.1" 200 45
10.244.0.1 - - ... "GET /favicon.ico HTTP/1.1" 404 196
10.244.0.1 - - ... "-" 408 -
10.244.0.1 - - ... "HEAD / HTTP/1.1" 200 -
10.244.0.1 - - ... "HEAD / HTTP/1.1" 200 -
10.244.0.1 - - ... "HEAD / HTTP/1.1" 200 -
10.244.0.1 - - ... "HEAD / HTTP/1.1" 200 -
10.244.0.1 - - ... "HEAD / HTTP/1.1" 200 -
10.244.0.1 - - ... "HEAD / HTTP/1.1" 200 -
10.244.0.1 - - ... "HEAD / HTTP/1.1" 200 -
10.244.0.1 - - ... "HEAD / HTTP/1.1" 200 -
10.244.0.1 - - ... "HEAD / HTTP/1.1" 200 -
10.244.0.1 - - ... "POST /boaform/admin/formLogin HTTP/1.1" 404 196

$ # The webserver container is running fine. Check the logs for other
# container (my-app).
$ kubectl logs my-deployment-fc94b7f98-m9z21 -c my-app

$ # No logs observed. The container could be starting or be in a transition
phase.
$ # So logs for the previous execution of this container can be checked
using the --previous flag:
$ kubectl logs my-deployment-fc94b7f98-m9z21 -c my-app --previous
<Some Logs from the container>
..
..
Started increasing memory
```

Log entries were made the previous time that the container was run. The existence of these entries suggests that the application did start, but it closed because of some issues.

Check the service associated with the deployment and try to curl the cluster IP of the service from inside the cluster to identify the issue:

```
Console

$ kubectl get svc # Check the service associated with deployment
NAME          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
AGE
kubernetes    ClusterIP   10.0.0.1       <none>        443/TCP
3h21m
my-deployment-service   LoadBalancer  10.0.136.71   20.62.x.x
80:30790/TCP   21m
```

The next step is to check the events of the pod by running the [kubectl describe](#) command:

```
Console

$ kubectl describe pod my-deployment-fc94b7f98-m9z21
Name:           my-deployment-fc94b7f98-m9z21
Namespace:      default
...
Labels:         app=my-pod
...
Containers:
  webserver:
  ...
  ...
  my-app:
    Container ID: containerd://a46e5062d53039d0d812c57c76b740f8d1ffb222de35203575bf8e4d10d6b51
    Image:          my-repo/my-image:latest
    Image ID:       docker.io/my-repo/my-image@sha256:edcc4bedc7b...
    State:          Running
    Started:        <Start Date>
    Last State:     Terminated
    Reason:         OOMKilled
    Exit Code:      137
    Ready:          True
    Restart Count:  44
    Limits:
      memory: 500Mi
    Requests:
      cpu: 250m
      memory: 500Mi
    ...
    ...
Events:
  Type  Reason  Age
  From      Message
```

```
--> Normal  Pulling  49m (x37 over 4h4m)    kubelet  Pulling image "my-  
repo/my-image:latest"  
Warning  BackOff  4m10s (x902 over 4h2m)  kubelet  Back-off restarting  
failed container
```

Observations:

- The exit code is 137. For more information about exit codes, see the [Docker run reference](#) and [Exit codes with special meanings](#).
- The termination reason is `OOMKilled`.
- The memory limit specified for the container is 500 Mi.

You can tell from the events that the container is being killed because it's exceeding the memory limits. When the container memory limit is reached, the application becomes intermittently inaccessible, and the container is killed and restarted.

Note

We recommend [configuring liveness, readiness, and startup probes](#) in your pod definition. Depending on your application's behavior, this configuration can help recover the application from unexpected issues. Be cautious when configuring liveness probes.

Solution

You can remove the memory limit and monitor the application to determine how much memory it actually needs. After you learn the memory usage, you can update the memory limits on the container. If the memory usage continues to increase, determine whether there's a memory leak in the application.

For more information about how to plan resources for workloads in Azure Kubernetes Service, see [resource management best practices](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Basic troubleshooting of cluster connection issues with the API server

07/08/2025

This article discusses connection issues to an Azure Kubernetes Service (AKS) cluster when you can't reach the cluster's API server through the Kubernetes cluster command-line tool ([kubectl](#)) or any other tool, such as using REST API through a programming language.

Prerequisites

- [Azure CLI](#).

Root cause and solutions

Connection issues to the API server can occur for many reasons, but the root cause is often related to an error with one of these items:

- Network
- Authentication
- Authorization

You can take these common troubleshooting steps to check the connectivity to the AKS cluster's API server:

1. Enter the following [az aks show](#) command in Azure CLI. This command gets the fully qualified domain name (FQDN) of your AKS cluster.

First, export your resource names to environment variables and add a random suffix to the resource group and cluster names for unique testing.

Azure CLI

```
export RANDOM_SUFFIX=$(head -c 3 /dev/urandom | xxd -p)
export RESOURCE_GROUP="my-aks-rg$RANDOM_SUFFIX"
export AKS_CLUSTER="myakscluster$RANDOM_SUFFIX"
az aks show --resource-group $RESOURCE_GROUP --name $AKS_CLUSTER --query fqdn
```

Results:

Output

```
"xxxxxxxx-xxxxxxx.hcp.eastus2.azmk8s.io"
```

- With the FQDN, check whether the API server is reachable from the client machine by using the name server lookup ([nslookup](#)), client URL ([curl ↗](#)), and [telnet](#) commands:

Replace `<cluster-fqdn>` with the actual FQDN returned from the previous step. For demonstration, we use a variable.

Bash

```
export CLUSTER_FQDN=$(az aks show --resource-group $RESOURCE_GROUP --name $AKS_CLUSTER --query fqdn -o tsv)

# Check if the DNS Resolution is working:
nslookup $CLUSTER_FQDN

# Then check if the API Server is reachable:
curl -k -Iv https://$CLUSTER_FQDN

# Test raw TCP connectivity (output will vary depending on environment)
timeout 5 telnet $CLUSTER_FQDN 443 || echo "Connection test completed"
```

- If the AKS cluster is private, make sure you run the command from a virtual machine (VM) that can access the AKS cluster's Azure Virtual Network. See [Options for connecting to the private cluster](#).
- If necessary, follow the steps in the troubleshooting article [Client IP address can't access the API server](#), so the API server adds your client IP address to the IP ranges it authorizes.
- Make sure the version of kubectl on your client machine isn't two or more minor versions behind the AKS cluster's version of that tool. To install the latest version of kubectl, run the [az aks install-cli](#) command in Azure CLI. You can then run [kubectl version ↗](#) command to check the version number of the new installation.

For example, on Linux you would run these commands:

shell

```
sudo az aks install-cli
kubectl version --client
```

For other client operating systems, use these [kubectl installation instructions ↗](#).

- If necessary, follow the steps in the troubleshooting article [Config file isn't available when connecting](#), so your Kubernetes configuration file (`config`) is valid and can be found at

connection time.

7. If necessary, follow the steps in the troubleshooting article [User can't get cluster resources](#), so you can list the details of your cluster nodes.
8. If you're using a firewall to control egress traffic from AKS worker nodes, make sure the firewall allows the [minimum required egress rules for AKS](#).
9. Make sure the [network security group that's associated with AKS nodes](#) allows communication on TCP port 10250 within the AKS nodes.

For other common troubleshooting steps, see [TCP time-outs when kubectl or other third-party tools connect to the API server](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Client IP address can't access the API server

Article • 10/08/2024

This article describes how to fix issues that occur when you can't connect to an Azure Kubernetes Service (AKS) cluster because your client IP address can't access the AKS API server.

Prerequisites

- [Azure CLI](#).
- The client URL ([curl ↗](#)) tool.

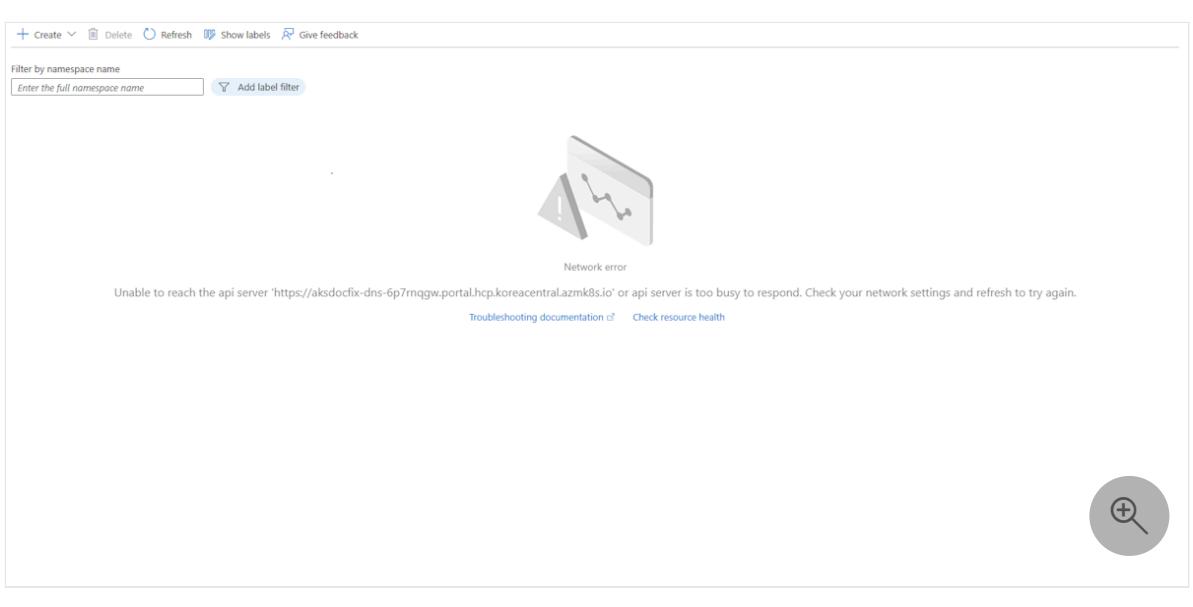
Symptoms

Azure portal

When you try to access Kubernetes resources such as namespaces and workloads from the Azure portal, you might encounter the following errors:

Network error

Unable to reach the api server 'https://<API-server-FQDN>' or api server is too busy to respond. Check your network settings and refresh to try again.



The screenshot shows a screenshot of the Azure portal interface. At the top, there's a navigation bar with options like '+ Create', 'Delete', 'Refresh', 'Show labels', and 'Give feedback'. Below the navigation bar, there's a search bar labeled 'Filter by namespace name' with a placeholder 'Enter the full namespace name' and a 'Add label filter' button. The main area displays a large error message: 'Network error' with a small icon of a computer monitor showing a line graph. Below the icon, the text reads: 'Unable to reach the api server 'https://aksdocfix-dns-6p7rnqgw.portal.hcp.koreacentral.azmk8s.io' or api server is too busy to respond. Check your network settings and refresh to try again.' At the bottom of the error message, there are two links: 'Troubleshooting documentation ↗' and 'Check resource health'. In the bottom right corner of the screenshot, there's a circular button with a magnifying glass icon and a plus sign inside it.

Cause

API server-authorized IP ranges may have been enabled on the cluster's API server, but the client's IP address wasn't included in the IP ranges. To check whether this feature has been enabled, see if the following `az aks show` command in Azure CLI produces a list of IP ranges:

Azure CLI

```
az aks show --resource-group <cluster-resource-group> \
--name <cluster-name> \
--query apiServerAccessProfile.authorizedIpRanges
```

Solution

Look at the cluster's API server-authorized ranges, and add your client's IP address within that range.

ⓘ Note

1. Do you access the API server from a corporate network where traffic is routed through a proxy server or firewall? Then ask your network administrator before you add your client IP address to the list of authorized ranges for the API server.
2. Also ask your cluster administrator before you add your client IP address, because there might be security concerns with adding a temporary IP address to the list of authorized ranges.

Azure portal

1. Navigate to the cluster from the Azure portal.
2. In the left menu, locate **Settings** and then select **Networking**.
3. On the **Networking** page, select the **Overview** tab.
4. Select **Manage** under **Resource settings**.
5. In the **Authorized IP ranges** pane, add your client IP address as shown in the following screenshot:

The screenshot shows the AKS Networking settings page. The left sidebar includes options like Diagnose and solve problems, Microsoft Defender for Cloud, Cost analysis, Resource visualizer, Namespaces, Workloads, Services and ingresses, Storage, Configuration, Custom resources, Events, Run command, Settings, Node pools, Cluster configuration, Application scaling, and Networking. The Networking option is highlighted with a red box. The main content area has tabs for Overview, Public access, and Virtual network integration. Under Overview, there's a Network profile section with Network configuration (Azure CNI Node Subnet, Migrate to Azure CNI Overlay), Pod CIDR (172.17.0/16), Service CIDR (10.0.0.10), DNS service IP (Azure), and Network policy (-). Resource settings include Public access to API server (Enabled), Load balancer (Standard), and Authorized IP ranges (Not enabled, Manage button). The Authorized IP ranges section shows a single entry: 0.2.0.0/32. There are Save and Cancel buttons at the bottom right.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?



Config file isn't available when connecting

Article • 09/16/2024

This article describes how to fix issues that occur when you can't connect to an Azure Kubernetes Service (AKS) cluster because of a missing or invalid *config* file.

Prerequisites

- [Azure CLI](#).
- The Kubernetes cluster command-line tool ([kubectl](#)). You can alternatively install kubectl by running the [az aks install-cli](#) command in Azure CLI.

Symptoms

During a cluster connection attempt, an error message similar to the following text appears:

Output

```
Unable to connect to the server: dial tcp [::1]:8080: connectex: No
connection could be made because the target machine actively refused it.

error: You must be logged in to the server (the server has asked for the
client to provide credentials)
```

Causes

The [kubectl tool](#) and other Kubernetes connection tools use a local configuration file named *config*. The *config* file contains authentication credentials and details to connect to the cluster. By default:

- The [az aks get-credentials](#) command in Azure CLI, which is used to get access credentials for a managed Kubernetes cluster, modifies the `~/.kube/config` file.
- The `kubectl` command uses the [kubeconfig \(kubectl configuration\) file](#) in the `$HOME/.kube` directory.

So what happens during an attempted Kubernetes session depends on the user who's running the `kubectl` command. If you've signed in as user A, and executed both

commands, here's what happens:

- The `az aks get-credentials` command tries to add the new kubeconfig parameters in the `C:\Users\A\.kube\config` file.
- The `kubectl` command tries to search the `C:\Users\A\.kube\config` file.

But for `kubectl`, if the pointer to the kubeconfig file has changed, the file that's used for accessing the cluster is supposed to be in a different location.

ⓘ Note

A kubeconfig file is a reference to a file that contains configuration parameters for accessing Kubernetes clusters. It doesn't necessarily refer to a file that's named `kubeconfig`.

The error occurs if one of the following scenarios occurs:

Cause 1: The `config` file doesn't exist

The `config` file doesn't exist on your machine.

Solution: Save the credentials

Load the `config` file by running the `az aks get-credentials` command in Azure CLI, which saves the credentials. If you don't want to use the default location, specify the `--file <config-file-location>` parameter with the location of `config` (for example, `~/Dir1/Dir2/config` or `C:\Dir1\Dir2\config`).

Azure CLI

```
az aks get-credentials --resource-group <cluster-resource-group> \
    --name <cluster-name> \
    [--file <config-file-location>]
```

Cause 2: The `config` file is in the wrong directory

The `config` file is on your machine, but it's in a different directory from where the `az aks get-credentials` command and/or the `kubectl` tool expects it to be.

Solution: Move the *config* file, save the credentials again, or change the KUBECONFIG environment variable

Take one or more of the following actions:

- Move the *config* file to the directory you want it to be in.
- Run the `az aks get-credentials` command. Specify the location you want if it isn't the default location.

Azure CLI

```
az aks get-credentials --resource-group <cluster-resource-group> \
--name <cluster-name> \
[--file <config-file-location>]
```

- Use one of the following options to change where kubectl looks for configuration parameters:
 - Modify the `KUBECONFIG` environment variable to point to the *config* file's current location. For more information, see [Set the KUBECONFIG environment variable](#).
 - Run the `kubectl config` command with the `--kubeconfig=<config-file-location>` parameter.

Cause 3: The *config* file has expired or is corrupted

The *config* file is on your machine. It's also in the expected directory for the `az aks get-credentials` command and the kubectl tool. But the file is expired or corrupted.

Solution: Reestablish the credentials

Reestablish the credentials, because the existing credentials might be expired or corrupted. In that case, you may run the `az aks get-credentials` command with the `--overwrite-existing` parameter.

Azure CLI

```
az aks get-credentials --resource-group <cluster-resource-group> \
--name <cluster-name> \
--overwrite-existing
```

```
--overwrite-existing \
[--file <config-file-location>]
```

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

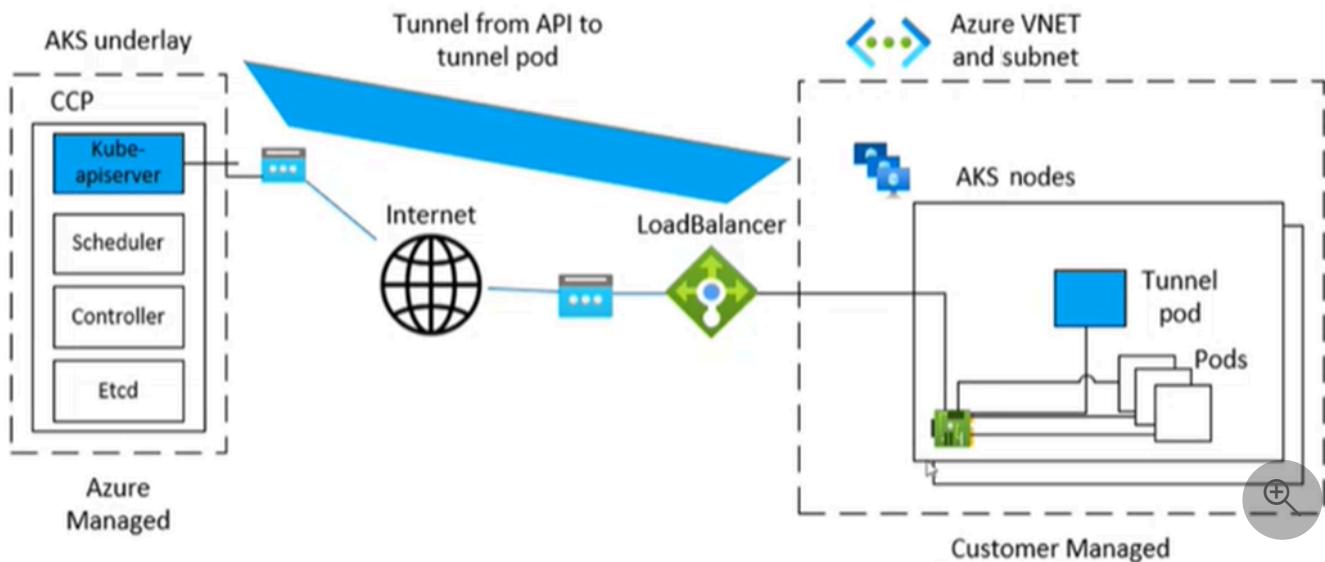
 Yes

 No

Tunnel connectivity issues

Article • 04/10/2025

Microsoft Azure Kubernetes Service (AKS) uses a specific component for tunneled, secure communication between the nodes and the control plane. The tunnel consists of a server on the control plane side and a client on the cluster nodes side. This article discusses how to troubleshoot and resolve issues that relate to tunnel connectivity in AKS.



! Note

Previously, the AKS tunnel component was [tunnel-front](#). It has now been migrated to the [Konnectivity service](#), an upstream Kubernetes component. For more information about this migration, see the [AKS release notes and changelog](#).

Prerequisites

- [Azure CLI](#)

Symptoms

You receive an error message that resembles the following examples about port 10250:

Error from server: Get "https://<aks-node-name>:10250/containerLogs/<namespace>/<pod-name>/<container-name>": dial tcp <aks-node-ip>:10250: i/o timeout

Error from server: error dialing backend: dial tcp <aks-node-ip>:10250: i/o timeout

Error from server: Get "https://<aks-node-name>:10250/containerLogs/<namespace>/<pod-name>/<container-name>": http: server gave HTTP response to HTTPS client

The Kubernetes API server uses port 10250 to connect to a node's kubelet to retrieve the logs. If port 10250 is blocked, the kubectl logs and other features will only work for pods that run on the nodes in which the tunnel component is scheduled. For more information, see [Kubernetes ports and protocols: Worker nodes](#).

Because the tunnel components or the connectivity between the server and client can't be established, functionality such as the following won't work as expected:

- Admission controller webhooks
- Ability of log retrieval (using the [kubectl logs](#) command)
- Running a command in a container or getting inside a container (using the [kubectl exec](#) command)
- Forwarding one or more local ports of a pod (using the [kubectl port-forward](#) command)

Cause 1: A network security group (NSG) is blocking port 10250

Note

This cause is applicable to any tunnel components that you might have in your AKS cluster.

You can use an [Azure network security group](#) (NSG) to filter network traffic to and from Azure resources in an Azure virtual network. A network security group contains security rules that allow or deny inbound and outbound network traffic between several types of Azure resources. For each rule, you can specify source and destination, port, and protocol. For more information, see [How network security groups filter network traffic](#).

If the NSG blocks port 10250 at the virtual network level, tunnel functionalities (such as logs and code execution) will work for only the pods that are scheduled on the nodes where tunnel pods are scheduled. The other pods won't work because their nodes won't be able to reach the tunnel, and the tunnel is scheduled on other nodes. To verify this state, you can test the connectivity by using [netcat](#) (`nc`) or telnet commands. You can run the [az vmss run-](#)

command `invoke` command to conduct the connectivity test and verify whether it succeeds, times out, or causes some other issue:

Azure CLI

```
az vmss run-command invoke --resource-group <infra-or-MC-resource-group> \
--name <virtual-machine-scale-set-name> \
--command-id RunShellScript \
--instance-id <instance-id> \
--scripts "nc -v -w 2 <ip-of-node-that-schedules-the-tunnel-component> 10250"
\
--output tsv \
--query 'value[0].message'
```

Solution 1: Add an NSG rule to allow access to port 10250

If you use an NSG, and you have specific restrictions, make sure that you add a security rule that allows traffic for port 10250 at the virtual network level. The following [Azure portal](#) image shows an example security rule:



Add inbound security rule

X

Source ⓘ

Service Tag



Source service tag * ⓘ

VirtualNetwork



Source port ranges * ⓘ

*

Destination ⓘ

Service Tag



Destination service tag ⓘ

VirtualNetwork



Service ⓘ

Custom



Destination port ranges * ⓘ

10250



Protocol

Any

TCP

UDP

ICMP

Action

Allow

Deny

Priority * ⓘ

100

Name *

Port_10250



If you want to be more restrictive, you can allow access to port 10250 at the subnet level only.

(!) Note

- The **Priority** field must be adjusted accordingly. For example, if you have a rule that denies multiple ports (including port 10250), the rule that's shown in the image should have a lower priority number (lower numbers have higher priority). For more information about **Priority**, see [Security rules](#).
- If you don't see any behavioral change after you apply this solution, you can re-create the tunnel component pods. Deleting these pods causes them to be re-created.

Cause 2: The Uncomplicated Firewall (UFW) tool is blocking port 10250

(!) Note

This cause applies to any tunnel component that you have in your AKS cluster.

[Uncomplicated Firewall](#) (UFW) is a command-line program for managing a [netfilter](#) firewall. AKS nodes use Ubuntu. Therefore, UFW is installed on AKS nodes by default, but UFW is disabled.

By default, if UFW is enabled, it will block access to all ports, including port 10250. In this case, it's unlikely that you can use Secure Shell (SSH) to [connect to AKS cluster nodes for troubleshooting](#). This is because UFW might also be blocking port 22. To troubleshoot, you can run the `az vmss run-command invoke` command to invoke a [ufw command](#) that checks whether UFW is enabled:

Azure CLI

```
az vmss run-command invoke --resource-group <infra-or-MC-resource-group> \
    --name <virtual-machine-scale-set-name> \
    --command-id RunShellScript \
    --instance-id <instance-id> \
    --scripts "ufw status" \
    --output tsv \
    --query 'value[0].message'
```

What if the results indicate that UFW is enabled, and it doesn't specifically allow port 10250? In this case, tunnel functionalities (such as logs and code execution) won't work for the pods that

are scheduled on the nodes that have UFW enabled. To fix the problem, apply one of the following solutions on UFW.

ⓘ Important

Before you use this tool to make any changes, review the [AKS support policy](#) (especially [node maintenance and access](#)) to prevent your cluster from entering into an unsupported scenario.

! Note

If you don't see any behavioral change after you apply a solution, you can re-create the tunnel component pods. Deleting these pods will cause them to be re-created.

Solution 2a: Disable Uncomplicated Firewall

Run the following `az vmss run-command invoke` command to disable UFW:

Azure CLI

```
az vmss run-command invoke --resource-group <infra-or-MC-resource-group> \
--name <virtual-machine-scale-set-name> \
--command-id RunShellScript \
--instance-id <instance-id> \
--scripts "ufw disable" \
--output tsv \
--query 'value[0].message'
```

Solution 2b: Configure Uncomplicated Firewall to permit access to port 10250

To force UFW to allow access to port 10250, run the following `az vmss run-command invoke` command:

Azure CLI

```
az vmss run-command invoke --resource-group <infra-or-MC-resource-group> \
--name <virtual-machine-scale-set-name> \
--command-id RunShellScript \
--instance-id <instance-id> \
--scripts "ufw allow 10250" \
```

```
--output tsv \  
--query 'value[0].message'
```

Cause 3: The iptables tool is blocking port 10250

! Note

This cause applies to any tunnel component that you have in your AKS cluster.

The [iptables](#) tool lets a system administrator configure the IP packet filter rules of a Linux firewall. You can configure the `iptables` rules to block communication on port 10250.

You can view the rules for your nodes to check whether port 10250 is blocked or the associated packets are dropped. To do this, run the following `iptables` command:

Bash

```
iptables --list --line-numbers
```

In the output, the data is grouped into several *chains*, including the `INPUT` chain. Each chain contains a table of rules under the following column headings:

- `num` (rule number)
- `target`
- `prot` (protocol)
- `opt`
- `source`
- `destination`

Does the `INPUT` chain contain a rule in which the target is `DROP`, the protocol is `tcp`, and the destination is `tcp dpt:10250`? If it does, `iptables` is blocking access to destination port 10250.

Solution 3: Delete the iptables rule that blocks access on port 10250

Run one of the following commands to delete the `iptables` rule that prevents access to port 10250:

Bash

```
iptables --delete INPUT --jump DROP --protocol tcp --source <ip-number> --destination-port 10250
```

Bash

```
iptables --delete INPUT <input-rule-number>
```

To address your exact or potential scenario, we recommend that you check the [iptables manual](#) by running the `iptables --help` command.

 **Important**

Before you use this tool to make any changes, review the [AKS support policy](#) (especially [node maintenance and access](#)) to prevent your cluster from entering into an unsupported scenario.

Cause 4: Egress port 1194 or 9000 isn't opened

 **Note**

This cause applies to only the `tunnel-front` and `aks-link` pods.

Are there any egress traffic restrictions, such as from an AKS firewall? If there are, port 9000 is required in order to enable correct functionality of the `tunnel-front` pod. Similarly, port 1194 is required for the `aks-link` pod.

Konnectivity relies on port 443. By default, this port is open. Therefore, you don't have to worry about connectivity issues on that port.

Solution 4: Open port 9000

Although `tunnel-front` has been moved to the [Konnectivity service](#), some AKS clusters still use `tunnel-front`, which relies on port 9000. Make sure that the virtual appliance or any network device or software allows access to port 9000. For more information about the required rules and dependencies, see [Azure Global required network rules](#).

Cause 5: Source Network Address Translation (SNAT) port exhaustion

➊ Note

This cause applies to any tunnel component that you have in your AKS cluster. However, it doesn't apply to [private AKS clusters](#). Source Network Address Translation (SNAT) port exhaustion can occur for public communication only. For private AKS clusters, the API server is inside the AKS virtual network or subnet.

If [SNAT port exhaustion](#) occurs (failed [SNAT ports](#)), the nodes can't connect to the API server. The tunnel container is on the API server side. Therefore, tunnel connectivity won't be established.

If the SNAT port resources are exhausted, the outbound flows fail until the existing flows release some SNAT ports. Azure Load Balancer reclaims the SNAT ports when the flow closes. It uses a four-minute idle time-out to reclaim the SNAT ports from the idle flows.

You can view the SNAT ports from either the AKS load balancer metrics or the service diagnostics, as described in the following sections. For more information about how to view SNAT ports, see [How do I check my outbound connection statistics?](#).

▼ AKS load balancer metrics

To use AKS load balancer metrics to view the SNAT ports, follow these steps:

1. In the [Azure portal](#), search for and select **Kubernetes services**.
2. In the list of Kubernetes services, select the name of your cluster.
3. In the menu pane of the cluster, find the **Settings** heading, and then select **Properties**.
4. Select the name that's listed under **Infrastructure resource group**.
5. Select the **kubernetes** load balancer.
6. In the menu pane of the load balancer, find the **Monitoring** heading, and then select **Metrics**.
7. For the metric type, select **SNAT Connection Count**.
8. Select **Apply splitting**.
9. Set **Split by** to **Connection State**.

▼ Service diagnostics

To use service diagnostics to view the SNAT ports, follow these steps:

1. In the [Azure portal](#), search for and select **Kubernetes services**.
2. In the list of Kubernetes services, select the name of your cluster.
3. In the menu pane of the cluster, select **Diagnose and solve problems**.
4. Select **Connectivity Issues**.
5. Under **SNAT Connection and Port Allocation**, select **View details**.
6. If necessary, use the **Time Range** button to customize the time frame.

Solution 5a: Make sure the application is using connection pooling

This behavior might occur because an application isn't reusing existing connections. We recommend that you don't create one outbound connection per request. Such a configuration can cause connection exhaustion. Check whether the application code is following best practices and using connection pooling. Most libraries support connection pooling. Therefore, you shouldn't have to create a new outbound connection per request.

Solution 5b: Adjust the allocated outbound ports

If everything is OK within the application, you'll have to adjust the allocated outbound ports. For more information about outbound port allocation, see [Configure the allocated outbound ports](#).

Solution 5c: Use a Managed Network Address Translation (NAT) Gateway when you create a cluster

You can set up a new cluster to use a Managed Network Address Translation (NAT) Gateway for outbound connections. For more information, see [Create an AKS cluster with a Managed NAT Gateway](#).

Cause 6: Konnectivity Agents performance issues with Cluster growth

As the cluster grows, the performance of Konnectivity Agents might degrade because of increased network traffic, more requests, or resource constraints.

! Note

This cause applies to only the `Konnectivity-agent` pods.

Solution 6: Cluster Proportional Autoscaler for Konnectivity Agent

To manage scalability challenges in large clusters, we implement the Cluster Proportional Autoscaler for our Konnectivity Agents. This approach aligns with industry standards and best practices. It ensures optimal resource usage and enhanced performance.

Why this change was made Previously, the Konnectivity agent had a fixed replica count that could create a bottleneck as the cluster grew. By implementing the Cluster Proportional Autoscaler, we enable the replica count to adjust dynamically, based on node-scaling rules, to provide optimal performance and resource usage.

How the Cluster Proportional Autoscaler works The Cluster Proportional Autoscaler work uses a ladder configuration to determine the number of Konnectivity agent replicas based on the cluster size. The ladder configuration is defined in the `konnectivity-agent-autoscaler` configmap in the `kube-system` namespace. Here is an example of the ladder configuration:

```
nodesToReplicas": [
  [1, 2],
  [100, 3],
  [250, 4],
  [500, 5],
  [1000, 6],
  [5000, 10]
]
```

This configuration makes sure that the number of replicas scales appropriately with the number of nodes in the cluster to provide optimal resource allocation and improved networking reliability.

How to use the Cluster Proportional Autoscaler? You can override default values by updating the `konnectivity-agent-autoscaler` configmap in the `kube-system` namespace. Here is a sample command to update the configmap:

Bash

```
kubectl edit configmap <pod-name> -n kube-system
```

This command opens the configmap in an editor to enable you to make the necessary changes.

What you should check

You have to monitor for Out Of Memory (OOM) kills on the nodes because misconfiguration of the Cluster Proportional Autoscaler can cause insufficient memory allocation for the Konnectivity agents. This misconfiguration occurs for the following key reasons:

High Memory Usage: As the cluster grows, the memory usage of Konnectivity agents can increase significantly. This increase can occur especially during peak loads or when handling large numbers of connections. If the Cluster Proportional Autoscaler configuration does not scale the replicas appropriately, the agents may run out of memory.

Fixed Resource Limits: If the resource requests and limits for the Konnectivity agents are set too low, they might not have enough memory to handle the workload, leading to OOM kills. Misconfigured Cluster Proportional Autoscaler settings can exacerbate this issue by not providing enough replicas to distribute the load.

Cluster Size and Workload Variability: The CPU and memory that are needed by the Konnectivity agents can vary widely depending on the size of the cluster and the workload. If the Cluster Proportional Autoscaler ladder configuration is not right-sized and adaptively resized for the cluster's usage patterns, it can cause memory overcommitment and OOM kills.

To identify and troubleshoot OOM kills, follow these steps:

1. Check for OOM Kills on nodes: Use the following command to check for OOM Kills on your nodes:

```
kubectl get events --all-namespaces | grep -i 'oomkill'
```

2. Inspect Node Resource Usage: Verify the resource usage on your nodes to make sure that they aren't running out of memory:

```
kubectl top nodes
```

3. Review Pod Resource Requests and Limits: Make sure that the Konnectivity agent pods have appropriate resource requests and limits set to prevent OOM Kills:

```
kubectl get pod <pod-name> -n kube-system -o yaml | grep -A5 "resources:"
```

4. Adjust Resource Requests and Limits: If necessary, adjust the resource requests and limits for the Konnectivity agent pods by editing the deployment:

```
kubectl edit deployment konnectivity-agent -n kube-system
```

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Troubleshoot "Forbidden" error when trying to access AKS cluster resources

07/09/2025

This article explains how to troubleshoot and resolve "Error from server (Forbidden)" errors that are related to Role-Based Access Control (RBAC) when you try to view Kubernetes resources in an Azure Kubernetes Service (AKS) cluster.

Prerequisites

The Kubernetes cluster command-line tool ([kubectl](#))

! Note

If you use [Azure Cloud Shell](#) to run shell commands, kubectl is already installed. If you use a local shell and already have [Azure CLI](#) installed, you can alternatively install kubectl by running the [az aks install-cli](#) command.

Symptoms

When you run `kubectl` commands to view details of a Kubernetes resource type, such as a deployment, pod, or worker node, you receive the following error message:

Output

```
$ kubectl get nodes
Error from server (Forbidden): nodes is forbidden: User "aaaaa11111-11aa-aa11-a1a1-11111aaaaa" cannot list resource "nodes" in API group "" at the cluster scope
```

Cause

This error indicates that you're trying to access Kubernetes resources by using a Microsoft Entra ID account that doesn't have the required role-based access control (RBAC) permissions.

Solution

Depending on the RBAC type that's configured for the cluster ([Kubernetes RBAC](#) or [Azure RBAC](#)), different solutions might apply. Run the following command to determine which RBAC

type the cluster is using:

Run the following command to determine which RBAC type your AKS cluster is using:

Bash

```
az aks show -g $RESOURCE_GROUP -n $CLUSTER_NAME --query aadProfile.enableAzureRbac
```

Results:

Output

```
false
```

- If the result is **null** or empty, the cluster doesn't have Azure AD integration enabled. See [Solving permission issues in local Kubernetes RBAC clusters](#).
- If the result is **false**, the cluster uses Kubernetes RBAC. See [Solving permission issues in Kubernetes RBAC-based AKS clusters](#).
- If the result is **true**, the cluster uses Azure RBAC. See [Solving permission issues in Azure RBAC-based AKS clusters](#).

Solving permissions issues in local Kubernetes RBAC clusters

If your cluster doesn't have Azure AD integration (result was null), it uses cluster admin credentials:

Bash

```
# Get admin credentials for full access
az aks get-credentials --resource-group $RESOURCE_GROUP --name $CLUSTER_NAME --admin

# Verify access
kubectl get nodes
```

Warning: Admin credentials provide full cluster access. Use carefully and consider enabling Azure AD integration for better security.

Solving permissions issues in Kubernetes RBAC-based AKS clusters

If the cluster uses Kubernetes RBAC, permissions for the user account are configured through the creation of RoleBinding or ClusterRoleBinding Kubernetes resources. For more information,

see [Kubernetes RBAC documentation](#).

Additionally, in Microsoft Entra ID integrated clusters, a ClusterRoleBinding resource is automatically created to grant the administrator access to the cluster to members of a pre-designated Microsoft Entra ID group.

To resolve the "Error from server (Forbidden)" error for a specific user, use one of the following methods.

Method 1: Create a custom RoleBinding or ClusterRoleBinding resource

You can create a custom RoleBinding or ClusterRoleBinding resource to grant the necessary permissions to the user (or a group of which the user is a member). For detailed steps, see [Use Kubernetes role-based access control with Microsoft Entra ID in Azure Kubernetes Service](#).

Method 2: Add the user to the pre-designated Microsoft Entra ID admin group

1. Retrieve the ID of the pre-designated Microsoft Entra ID admin group. To do this, run the following command:

Bash

```
az aks show -g $RESOURCE_GROUP -n $CLUSTER_NAME --query  
aadProfile.adminGroupObjectIDs
```

Results:

Output

```
[  
  "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"  
]
```

2. Add the user to the pre-designated Microsoft Entra ID admin group by using the group ID that you retrieved in the previous step. For more detailed steps, see [Add members or owners of a group](#).

Solving permissions issues in Azure RBAC-based AKS clusters

If the cluster uses Azure RBAC, permissions for users are configured through the creation of [Azure role assignments](#).

AKS provides a set of built-in roles that can be used to create role assignments for the Microsoft Entra ID users or groups to give them access to Kubernetes objects in a specific namespace or at cluster scope. For detailed steps to assign built-in roles to users or groups in Azure RBAC-based clusters, see [AKS built-in roles](#).

Alternatively, you can create your own custom Azure role definitions to provide a more granular management of permissions over specific types of Kubernetes objects and operations. For detailed guidance to create and assign custom roles to users and groups in Azure RBAC-based clusters, see [Create custom roles definitions](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Can't access the cluster API server when using authorized IP ranges

Article • 03/26/2025

This article discusses how to resolve a scenario in which you can't use authorized IP address ranges to access the API server for a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

If you try to create or manage resources in an AKS cluster, you can't access the cluster API server. When you run `kubectl`, you receive the following error message:

Unable to connect to the server: dial tcp x.x.x.x:443: i/o timeout

Cause

You configured your AKS cluster to access the cluster API server by using authorized IP address ranges that your computer can't access.

Solution

Make sure that when you run the `az aks create` or `az aks update` command in [Azure CLI](#), the `--api-server-authorized-ip-ranges` parameter includes the IP addresses or IP address ranges of the automation, development, or tooling systems that are being used.

More information

- [How to find my IP to include in --api-server-authorized-ip-ranges?](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Provide product feedback ↗

Basic troubleshooting of outbound connections from an AKS cluster

07/24/2025

This article discusses how to do basic troubleshooting of outbound connections from a Microsoft Azure Kubernetes Service (AKS) cluster and identify faulty components.

Prerequisites

- The Kubernetes [kubectl](#) tool, or a similar tool to connect to the cluster. To install kubectl by using [Azure CLI](#), run the `az aks install-cli` command.
- The [apt-get](#) command-line tool for handling packages.
- The Client URL ([cURL](#)) tool, or a similar command-line tool.
- The `nslookup` command-line ([dnsutils](#)) tool for checking DNS resolution.

Scenarios for outbound traffic in Azure Kubernetes Service

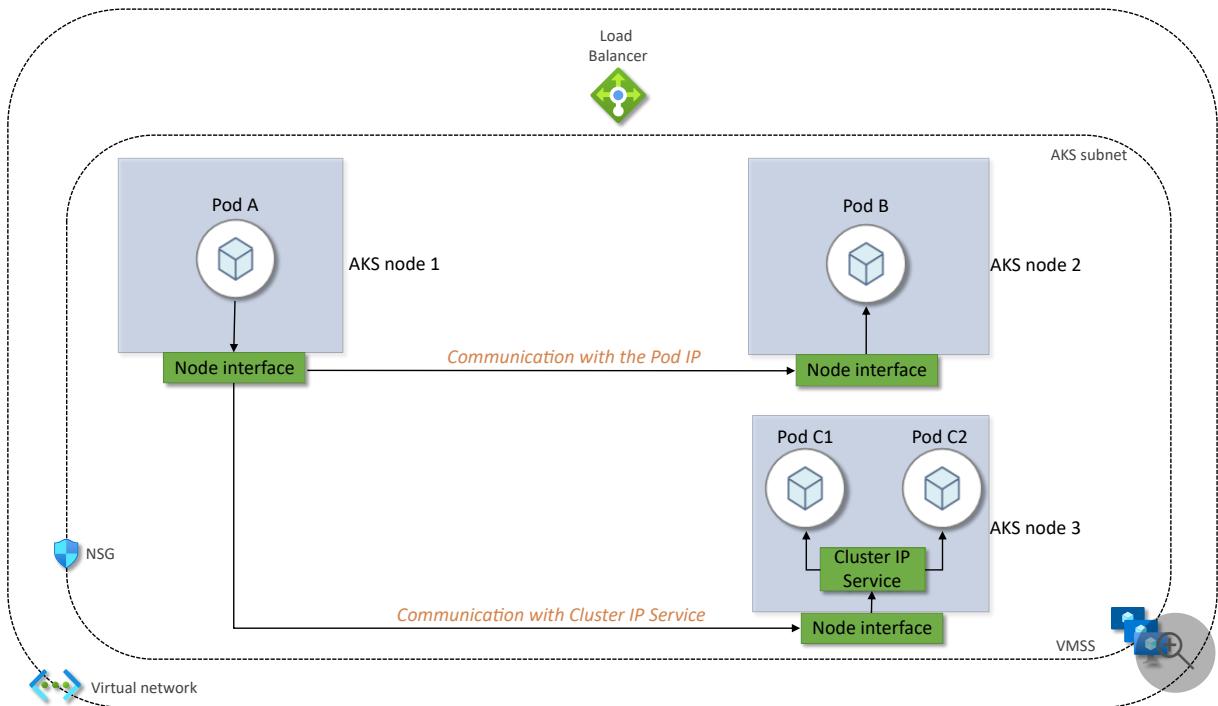
Traffic that originates from within the AKS cluster, whether it's from a pod or a worker node, is considered as outbound traffic from the cluster. If there's an issue in the outbound flow for an AKS cluster, before you troubleshoot, first look at the scenarios for outbound traffic flow.

The outbound traffic from an AKS cluster can be classified into the following categories:

1. [Traffic to a pod or service in the same cluster \(internal traffic\)](#).
2. Traffic to a network resource or endpoint in the same virtual network or a different virtual network that uses virtual network peering.
3. Traffic to an on-premises environment through a VPN connection or an Azure ExpressRoute connection.
4. [Traffic outside the AKS network through Azure Load Balancer \(public outbound traffic\)](#).
5. [Traffic outside the AKS network through Azure Firewall or a proxy server \(public outbound traffic\)](#).

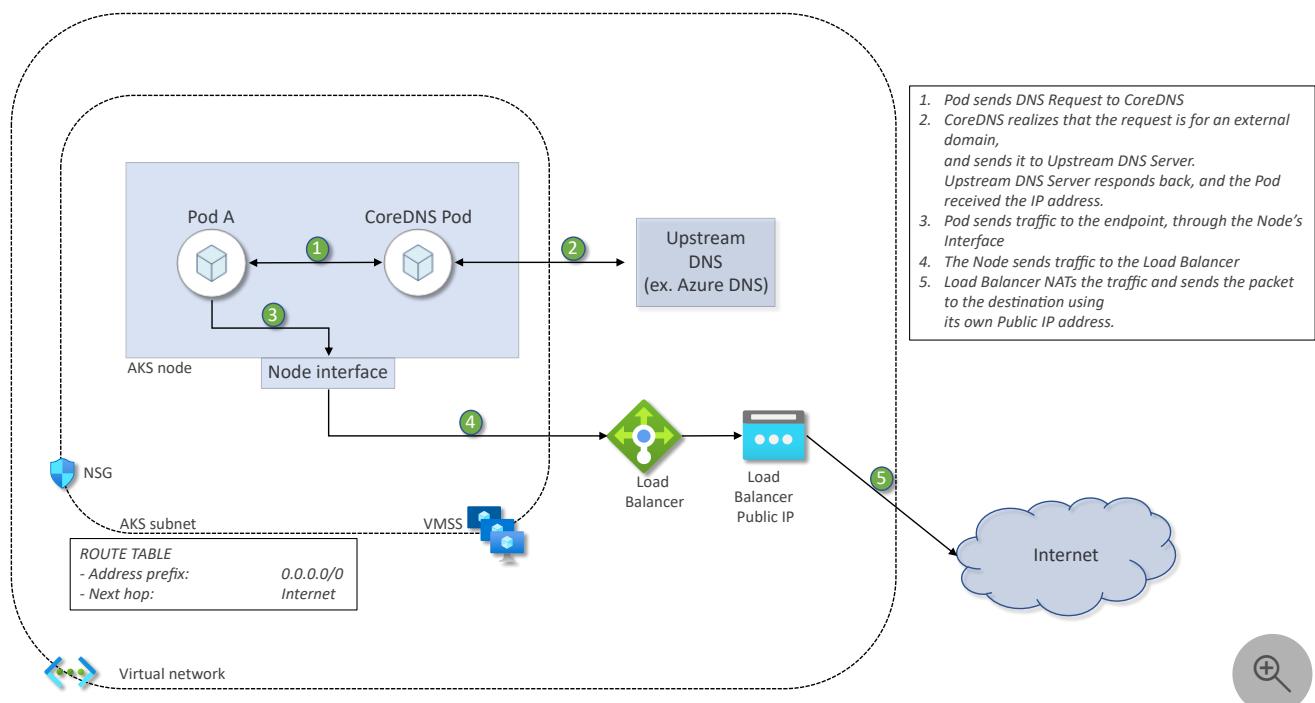
Internal traffic

A basic request flow for internal traffic from an AKS cluster resembles the flow shown in the following diagram.



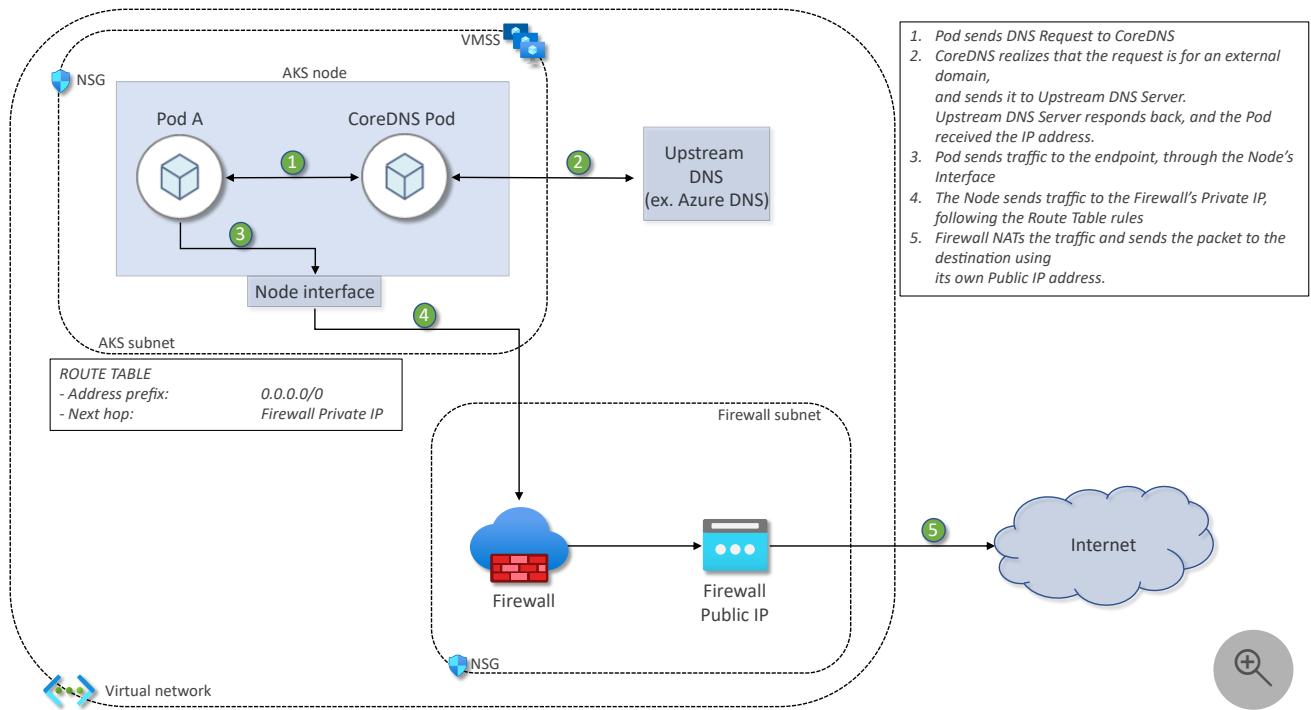
Public outbound traffic through Azure Load Balancer

If the traffic is for a destination on the internet, the default method is to send the traffic through the Azure Load Balancer.



Public outbound traffic through Azure Firewall or a proxy server

In some cases, the egress traffic has to be filtered, and it might require Azure Firewall.



A user might want to add a proxy server instead of a firewall, or set up a NAT gateway for egress traffic. The basic flow remains the same as shown in the diagram.

It's important to understand the nature of egress flow for your cluster so that you can continue troubleshooting.

Considerations when troubleshooting

Check your network resources within traffic flow

When you troubleshoot outbound traffic in AKS, it's important to know what network resources are present (that is, the hops through which the traffic passes). Here, the network resource could be one of the following components:

- Azure Load Balancer
- Azure Firewall or a custom firewall
- A network address translation (NAT) gateway
- A proxy server
- Network security group (NSG)
- Network policy

The flow could also differ based on the destination. For example, internal traffic (that is, within the cluster) doesn't go through the external network resources and only uses the cluster

networking. For public outbound traffic, determine which network resources are implemented for your cluster.

Check outbound connectivity path and blockers with Azure Virtual Network Verifier (Preview)

To check where traffic is blocked within your network resources to specific endpoints (for example, `mcr.microsoft.com`), you can use the [Azure Virtual Network Verifier \(Preview\)](#) tool. By running a connectivity analysis, you can visualize the hops within the traffic flow and any misconfigurations within Azure networking resources that are blocking traffic. We recommend using the Virtual Network Verifier tool as a first step in troubleshooting outbound connectivity issues to isolate the issue and detect problematic network configuration. For more instructions, [check if Azure network resources are blocking traffic to the endpoint using Azure Virtual Network Verifier \(Preview\)](#).

Manual troubleshooting

For manual troubleshooting, we recommend you check the following items:

- The source and the destination for the request.
- The hops in between the source and the destination.
- The request-response flow.
- The hops enhanced by extra security layers, such as the following layers:
 - Firewall
 - Network security group (NSG)
 - Network policy

To identify a problematic hop, [check the HTTP response codes before and after it](#). These codes are useful to identify the nature of the issue. The codes are especially helpful in scenarios in which the application responds to HTTP requests. To check whether the packets arrive properly in a specific hop, you can proceed with packet captures.

Take packet captures from the client and server

If other troubleshooting steps don't provide any conclusive outcome, take packet captures from the client and server. Packet captures are also useful when non-HTTP traffic is involved between the client and server. For more information about how to collect packet captures for AKS environment, see the following articles in the data collection guide:

- Capture a TCP dump from a Linux node in an AKS cluster
- Capture a TCP dump from a Windows node in an AKS cluster
- Capture TCP packets from a pod on an AKS cluster

Troubleshooting checklists

For basic troubleshooting for egress traffic from an AKS cluster, follow these steps:

1. [Check if Azure network resources are blocking traffic to the endpoint using Azure Virtual Network Verifier \(Preview\).](#)
2. [Make sure that the Domain Name System \(DNS\) resolution for the endpoint works correctly.](#)
3. Make sure that you can reach the endpoint through an IP address.
4. [Make sure that you can reach the endpoint from another source.](#)
5. [Make sure that the endpoint is working.](#)
6. [Check whether the cluster can reach any other external endpoint.](#)
7. [Check whether a network policy is blocking the traffic.](#)
8. [Check whether an NSG is blocking the traffic.](#)
9. Check whether a firewall or proxy is blocking the traffic.
10. Check whether the AKS service principal or managed identity has the required [AKS service permissions](#) to make the network changes to Azure resources.

 **Note**

Assumes no service mesh when you do basic troubleshooting. If you use a service mesh such as Istio, it produces unusual outcomes for TCP based traffic.

Check if Azure network resources are blocking traffic to the endpoint

To determine if traffic is blocked to the endpoint due to Azure network resources, run a connectivity analysis from your AKS cluster nodes to the endpoint using the [Azure Virtual Network Verifier \(Preview\)](#) tool. The connectivity analysis covers the following resources:

- Azure Load Balancer
- Azure Firewall
- A network address translation (NAT) gateway
- Network security group (NSG)
- Network policy
- User defined routes (route tables)
- Virtual network peering

Note

Azure Virtual Network Verifier (Preview) can't access any external or third-party networking resources, such as a custom firewall. If the connectivity analysis doesn't detect any blocked traffic, we recommend that you perform a manual check of any external networking to cover all hops in the traffic flow.

Currently, clusters using Azure CNI Overlay aren't supported for this feature. Support for CNI Overlay is planned for August 2025.

1. Navigate to your cluster in the Azure portal. In the sidebar, navigate to the Settings -> Node pools blade.
2. Identify the nodepool you want to run a connectivity analysis from. Click on the nodepool to select it as the scope.
3. Select "Connectivity analysis (Preview)" from the toolbar at the top of the page. If you don't see it, click on the three dots "..." in the toolbar at the top of the page to open the expanded menu.
 image
4. Select a Virtual Machine Scale Set (VMSS) instance as the source. The source IP addresses are populated automatically.
5. Select a public domain name/endpoint as the destination for the analysis, one example is `mcr.microsoft.com`. The destination IP addresses are also populated automatically.
6. Run the analysis and wait up to 2 minutes for the results. In the resulting diagram, identify the associated Azure network resources and where traffic is blocked. To view the detailed analysis output, click on the "JSON output" tab or click into the arrows in the diagram.

Check that the Domain Name Service (DNS) resolution for the endpoint works correctly

You can run a DNS lookup to the endpoint by running a debugging pod on one of your AKS nodes. If the issue is isolated to a specific problematic pod or namespace, run the DNS lookup from within the same namespace where you notice the problem.

If you can't run the [kubectl exec](#) command to connect to an existing pod, you can [start a test pod in the same namespace as the problematic pod](#) to run the tests.

! Note

If the DNS resolution or egress traffic doesn't let you install the necessary network packages, you can use the `rishashi/ubuntu-netutil:1.0` docker image. In this image, the required packages are already installed.

Example procedure for checking DNS resolution

1. Start a test pod in the problematic namespace:

Bash

```
kubectl run -it --rm aks-ssh --namespace <namespace> --image=debian:stable --overrides='{"spec": { "nodeSelector": {"kubernetes.io/os": "linux"}}}'
```

After the test pod is running, you'll gain access to the pod.

2. Run the following `apt-get` commands to install other tool packages:

Bash

```
# Update and install tool packages
apt-get update && apt-get install -y dnsutils curl
```

3. After the packages are installed, run the `nslookup` command to test the DNS resolution to the endpoint:

Console

```
$ nslookup microsoft.com # Microsoft.com is used as an example
Server:      <server>
Address:     <server IP address>#53
...
...
Name:   microsoft.com
Address: 20.53.203.50
```

4. Try the DNS resolution from the upstream DNS server directly. This example uses Azure DNS:

Console

```
$ nslookup microsoft.com 168.63.129.16
Server:      168.63.129.16
Address:     168.63.129.16#53
...
...
Address: 20.81.111.85
```

Sometimes, there's a problem with the endpoint itself rather than a cluster DNS. In such cases, consider the following checks:

1. Check whether the desired port is open on the remote host:

```
Bash

curl -Ivm5 telnet://microsoft.com:443
```

2. Check the HTTP response code:

```
Bash

curl -Ivm5 https://microsoft.com
```

3. Check whether you can connect to any other endpoint:

```
Bash

curl -Ivm5 https://kubernetes.io
```

To verify that the endpoint is reachable and DNS is functioning from the node hosting the problematic pod, follow these steps:

1. Enter the node hosting the problematic pod using the debug pod. For more information, see [Connect to Azure Kubernetes Service \(AKS\) cluster nodes for maintenance or troubleshooting](#).
2. Test the DNS resolution to the endpoint:

```
Console

$ nslookup microsoft.com
Server:      168.63.129.16
Address:     168.63.129.16#53

Non-authoritative answer:
Name:  microsoft.com
Address: 20.112.52.29
```

```
Name: microsoft.com
Address: 20.81.111.85
Name: microsoft.com
Address: 20.84.181.62
Name: microsoft.com
Address: 20.103.85.33
Name: microsoft.com
Address: 20.53.203.50
```

3. Check the `resolv.conf` file to determine whether the expected name servers are added:

```
Bash

cat /etc/resolv.conf
cat /run/systemd/resolve/resolv.conf
```

Example procedure for checking DNS resolution of a Windows pod

1. Run a test pod in the Windows node pool:

```
Bash

# For a Windows environment, use the Resolve-DnsName cmdlet.
kubectl run dnsutil-win --image='mcr.microsoft.com/windows/servercore:ltsc2022' --overrides='{"spec": {"nodeSelector": {"kubernetes.io/os": "windows"}}}' -- powershell "Start-Sleep -s 3600"
```

2. Run the `kubectl exec` command to connect to the pod by using PowerShell:

```
Bash

kubectl exec -it dnsutil-win -- powershell
```

3. Run the `Resolve-DnsName` cmdlet in PowerShell to check whether the DNS resolution is working for the endpoint:

```
Console

PS C:\> Resolve-DnsName www.microsoft.com

Name          Type   TTL    Section      NameHost
----          ----   ---    -----      -----
www.microsoft.com      CNAME  20     Answer      www.microsoft.com-c-
3.edgekey.net
www.microsoft.com-c-3.edgekey. CNAME  20     Answer      www.microsoft.com-c-
3.edgekey.net.globalredir.akadns.net
```

```
net
www.microsoft.com-c-3.edgekey. CNAME 20      Answer
e13678.dscb.akamaiedge.net
net.globalredir.akadns.net

Name      : e13678.dscb.akamaiedge.net
QueryType : AAAA
TTL       : 20
Section   : Answer
IP6Address : 2600:1408:c400:484::356e

Name      : e13678.dscb.akamaiedge.net
QueryType : AAAA
TTL       : 20
Section   : Answer
IP6Address : 2600:1408:c400:496::356e

Name      : e13678.dscb.akamaiedge.net
QueryType : A
TTL       : 12
Section   : Answer
IP4Address : 23.200.197.152
```

In one unusual scenario that involves DNS resolution, the DNS queries get a correct response from the node but fail from the pod. For this scenario, you might consider [checking DNS resolution failures from inside the pod but not from the worker node](#). If you want to inspect DNS resolution for an endpoint across the cluster, you can consider [checking DNS resolution status across the cluster](#).

If the DNS resolution is successful, continue to the network tests. Otherwise, verify the DNS configuration for the cluster.

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Errors after restricting egress traffic in AKS

Article • 03/31/2025

This article discusses how to troubleshoot issues that occur after you restrict egress traffic for cluster nodes in Microsoft Azure Kubernetes Service (AKS).

Symptoms

Certain commands of the [kubectl](#) command-line tool don't work correctly, or you experience errors when you create an AKS cluster or scale a node pool.

Cause

When you restrict egress traffic from an AKS cluster, your settings must comply with required Outbound network and FQDN (fully qualified domain names) rules for AKS clusters. If your settings are in conflict with any of these rules, the egress traffic restriction issues occur.

Solution

Verify that your configuration doesn't conflict with any of the [required Outbound network and FQDN \(fully qualified domain names\) rules for AKS clusters](#) for the following items:

- Outbound ports
- Network rules
- FQDNs
- Application rules

Check for conflicts with the rules that might occur in the NSG (network security group), firewall, or appliance that AKS traffic passes through according to the configuration.

Note

The AKS outbound dependencies are almost entirely defined by using FQDNs. These FQDNs don't have static addresses behind them. The lack of static addresses means that you can't use NSGs to restrict outbound traffic from an AKS cluster. Additionally, scenarios that allow only IPs that are obtained from required FQDNs

after all deny in NSG are not enough to restrict outbound traffic. Because the IPs are not static, issues might occur later.

More information

- Limit network traffic with Azure Firewall in AKS clusters

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

Troubleshoot SNAT port exhaustion on Azure Kubernetes Service nodes

Article • 10/12/2024

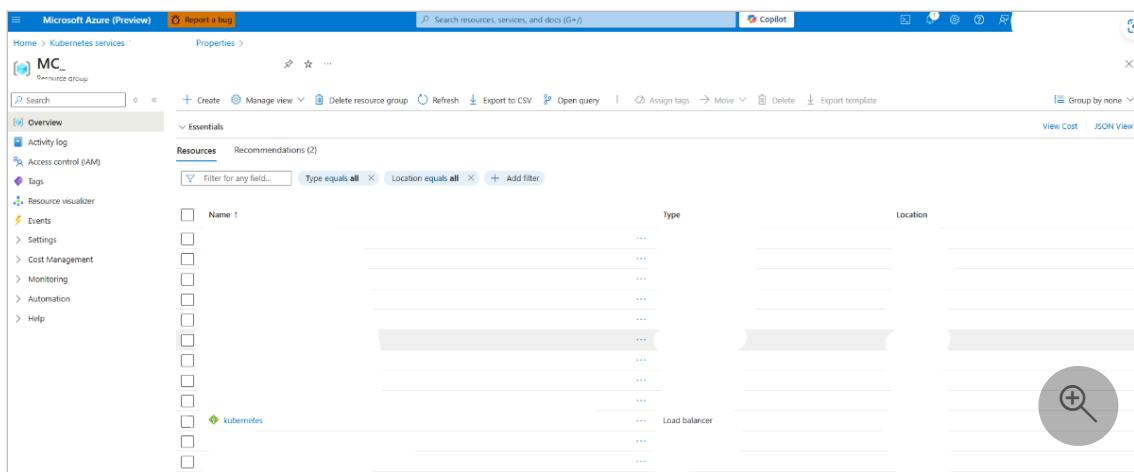
This article helps you find and troubleshoot Azure Kubernetes Service (AKS) nodes that experience Source Network Address Translation (SNAT) port exhaustion.

ⓘ Note

- To troubleshoot SNAT port exhaustion on AKS nodes in an AKS cluster running [Kubernetes jobs](#), perform the following steps only when the jobs are actively running on the AKS nodes.
- To learn more about SNAT ports and their allocation per virtual machine, see [What SNAT ports are](#).

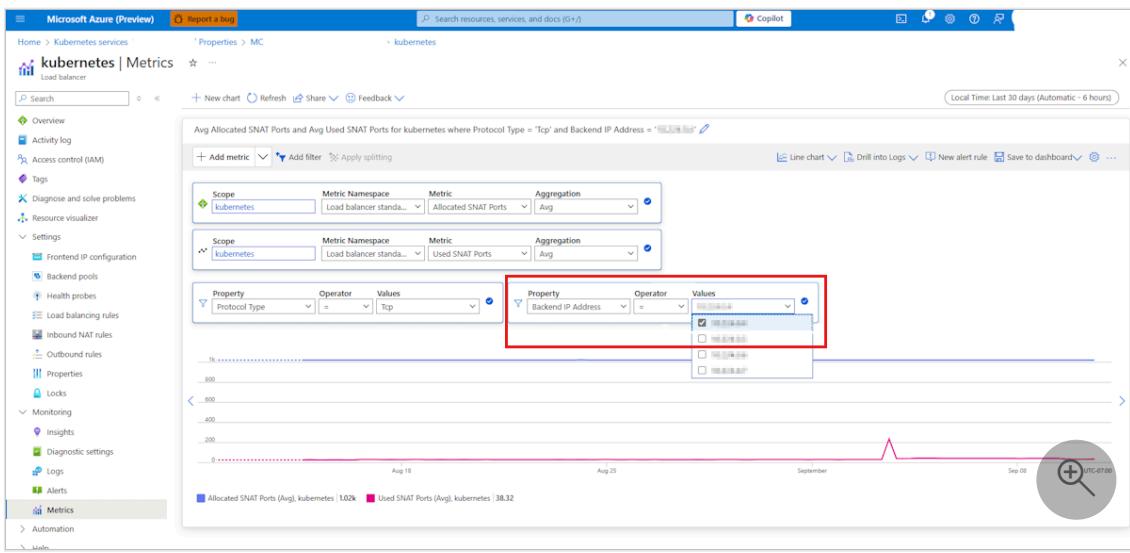
Step 1: Locate the node that experiences SNAT port exhaustion

1. Get the IP address of the AKS node that experiences active SNAT port exhaustion from the Azure portal.
 - a. Locate the default Kubernetes load balancer by navigating to your AKS cluster's resource group.



The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes 'Microsoft Azure (Preview)', 'Report a bug', 'Properties >', 'Search resources, services, and docs (F1)', 'Copilot', and various icons. Below the navigation is a breadcrumb trail: 'Home > Kubernetes services'. On the left, there's a sidebar with links like 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Resource visualizer', 'Events', 'Settings', 'Cost Management', 'Monitoring', 'Automation', and 'Help'. The main content area is titled 'Resources' and shows 'Recommendations (2)'. A search bar at the top right contains the text 'kubernetes'. The table below lists resources with columns for 'Name', 'Type', and 'Location'. One entry is highlighted: 'Load balancer' located in 'kubernetes'.

- b. Locate the AKS node that's experiencing SNAT port exhaustion by [checking SNAT port usage and allocation](#) on the load balancer metrics page. The **Values** drop-down list shows node IP addresses.



2. Connect to your AKS cluster and use the node IP address to get the node name by running the following `kubectl` command:

Console

```
kubectl get nodes -o wide | grep <node IP>
```

Step 2: Locate the Linux pod that has high outbound connections

! Note

- [Tcptracer](#) is one of the [BPF Compiler Collection \(BCC\) tools](#) that are pre-installed on Linux nodes. It allows you to trace TCP established connections (`connect()`, `accept()`, and `close()`). You can use it to find high outbound connections from the source IP address and network namespace (netns) of a pod.
- To access the BCC tools, use [kubectl node-shell](#) only.
- All the following commands in this section are run as the root user on a Linux node that has the BCC tools installed.

1. On the Linux node that experiences SNAT port exhaustion, install [kubectl node-shell](#):

Bash

```
curl -LO https://github.com/kvaps/kubectl-node-shell/raw/master/kubectl-node_shell
```

```
chmod +x ./kubectl-node_shell  
sudo mv ./kubectl-node_shell /usr/local/bin/kubectl-node_shell
```

2. Use SSH to connect to the node that experiences SNAT port exhaustion and use `tcptracer` to trace TCP established connections:

Bash

```
kubectl node-shell <node name>  
cd /usr/share/bcc/tools  
/usr/share/bcc/tools# python3 tcptracer -t4v
```

Here's a command output example:

Output

```
Tracing TCP established connections. Ctrl-C to end.  
TIME(ns)      TYPE        PID  COMM          IP SADDR  
DADDR          SPORT       DPORT  NETNS  
0             connect     18627 curl           4  1.2.3.4  
5.6.7.8       53746     80    4026532785  
3xxx9         close      18627 curl           4  1.2.3.4  
5.6.7.8       53746     80    4026532785  
1xxxxx4       connect     18629 curl           4  1.2.3.4  
9.10.11.12    35686     80    4026532785  
2xxxxx9       close      18629 curl           4  1.2.3.4  
9.10.11.12    35686     80    4026532785  
4xxxxx5       connect     18631 curl           4  1.2.3.4  
9.10.11.12    35688     80    4026532785  
4xxxxx8       close      18631 curl           4  1.2.3.4  
9.10.11.12    35688     80    4026532785  
7xxxxx3       connect     18633 curl           4  1.2.3.4  
13.14.15.16   35690     80    4026532785  
9xxxxx7       close      18633 curl           4  1.2.3.4  
13.14.15.16   35690     80    4026532785
```

3. Write the previous command output to a log file, and then sort the output to review a list of high connections:

Bash

```
python3 tcptracer -t4v > log  
head -n +2 log | tail -n 1 | awk '{print "Count", $6, $10}'; awk '{print  
$6, $10}' log | sort | uniq -c | sort -nrk 1 | column -t
```

Here's a command output example:

Output

Count	SADDR	NETNS
387	1.2.3.4	4026532785
8	11.22.33.44	4026532184
8	55.66.77.88	4026531992

4. Map the IP address with the most connections from the previous output to a pod.
If it doesn't work, you can continue.
5. Note the `SADDR` or `NETNS` value with the most connections from the previous output, and then run the following [lsns](#) command to map it to a PID. Lsns is a Linux tool that lists namespaces and maps namespaces to PIDs in the Linux process tree.

Bash

```
lsns -t net
```

Here's a command output example:

Output

NS	TYPE	NPROCS	PID	USER	COMMAND
4026532785	net	3	19832	root	bash

6. Map the previous PID to a containerd process by using [pstree](#). Pstree is a Linux tool that lists processes in a tree format for readability.

Bash

```
pstree -aps 19832
```

Here's a command output example:

Output

```
systemd,1
`-containerd-shim,20946 -namespace k8s.io -id 2xxx...  
...
```

Note the first five characters of the containerd `-id` in the command output. It will be used in step 7.

7. Map the previous containerd `-id` value to a POD ID by using [crictl](#). Crictl provides a CLI for CRI-compatible container runtimes.

```
Bash
```

```
crlctl ps -a
```

Here's a command output example:

```
Output
```

CONTAINER POD	IMAGE	CREATED	STATE	NAME	ATTEMPT	POD ID
6b5xxxxxb 2xxxxxxxxxf	fbxxxxx1 nginx	6 hours ago	Running	ubuntu	0	

Use the first five characters of the previous containerd `-id` value to match a POD ID.

- Get all pods running on the node and use the previous POD ID to match the pod that has high outbound connections from the command output:

```
Bash
```

```
kubectl get pods --all-namespaces -o wide --field-selector spec.nodeName=<nodename>
```

Step 3: Find all outbound network connections made by the application

For a Linux pod

- Execute into the pod that's identified as having high outbound connections in [Step 2](#) by using one of the following commands:

- Bash

```
kubectl exec -it <pod name> -n <namespace> /bin/bash
```

- Bash

```
kubectl exec -it <pod name> -n <namespace> /bin/sh
```

2. Install the netstat command-line tool in the pod by running the following command. Netstat is a network troubleshooting tool for admins only.

- On Debian, Ubuntu, or Linux Mint

```
Bash
```

```
apt update  
apt install net-tools
```

- On RHEL, CentOS, Fedora, AlmaLinux, or Rocky Linux

```
Bash
```

```
yum update  
yum install net-tools
```

- On Gentoo Linux

```
Bash
```

```
emerge -a sys-apps/net-tools
```

- On Alpine Linux

```
Bash
```

```
apk add net-tools
```

- On Arch Linux

```
Bash
```

```
pacman -S net-tools
```

- On OpenSUSE

```
Bash
```

```
zypper install net-tools
```

3. Once netstat is installed in the pod, run the following command:

```
Bash
```

```
netstat -ptn | grep -i established
```

Here's a command output example:

Output

```
tcp      0      0 10.x.x.x:xxxx      20.x.x.x:443
ESTABLISHED xxxxx3/telnet
```

In the command output, the local address is the pod's IP address, and the foreign address is the IP to which the application connects. The public IP connections in the `ESTABLISHED` state are the connections that utilize SNAT. Ensure that you count only the connections in the `ESTABLISHED` state to public IP addresses and ignore any connections in the `ESTABLISHED` state to private IP addresses.

Repeat the steps in this section for all other pods running on the node. The pod that has the most connections in the `ESTABLISHED` state to public IP addresses hosts the application that causes SNAT port exhaustion on the node. Work with your application developers to tune the application for improved network performance using the recommendations mentioned in [Design connection-efficient applications](#). After implementing the recommendations, verify that you see less SNAT port exhaustion.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Troubleshoot connection issues to pods or services within an AKS cluster (internal traffic)

Article • 09/05/2024

This article discusses how to troubleshoot connection issues to pods or services as internal traffic from within the same Microsoft Azure Kubernetes Services (AKS) cluster.

Prerequisites

- The Kubernetes [kubectl](#) tool, or a similar tool to connect to the cluster. To install kubectl by using [Azure CLI](#), run the `az aks install-cli` command.
- The [apt-get](#) command-line tool for handling packages.
- The Client URL ([cURL](#)) tool, or a similar command-line tool.
- The [Netcat](#) (`nc`) command-line tool for TCP connections.

Troubleshooting checklist

Step 1: Set up the test pod and remote server port

Set up the test pod and make sure that the required port is open on the remote server. From within the source pod (or a test pod that's in the same namespace as the source pod), follow these steps:

1. Start a test pod in the cluster by running the [kubectl run](#) command:

Bash

```
kubectl run -it --rm aks-ssh --namespace <namespace> --  
image=debian:stable
```

2. After you gain access to the pod, run the following [apt-get](#) commands to install the DNS Utils, cURL, and Netcat packages:

Bash

```
apt-get update -y  
apt-get install dnsutils -y  
apt-get install curl -y  
apt-get install netcat-openbsd -y
```

3. After the packages are installed, run the following cURL command to test the connectivity to the IP address of the pod:

Bash

```
curl -Iv http://<pod-ip-address>:<port>
```

4. Run the Netcat command to check whether the remote server opened the required port:

Bash

```
nc -z -v <endpoint> <port>
```

Step 2: View operational information about pods, containers, the Kubernetes services, and endpoints

Using kubectl and cURL at the command line, follow these steps to check that everything works as expected:

1. Verify that the destination pod is up and running:

Bash

```
kubectl get pods -n <namespace-name>
```

If the destination pod is working correctly, the pod status is shown as `Running`, and the pod is shown as `READY`.

Output

NAME	READY	STATUS	RESTARTS	AGE
my-other-pod	1/1	Running	0	44m
my-pod	1/1	Running	0	44m

2. Search the pod logs for access errors:

```
Bash
```

```
kubectl logs <pod-name> -n <namespace-name>
```

3. Search the pod logs for an individual container in a multicontainer pod:

```
Bash
```

```
kubectl logs <pod-name> -n <namespace-name> -c <container-name>
```

4. If the application that's inside the pod restarts repeatedly, view pod logs of a previous container instance to get the exit messages:

```
Bash
```

```
kubectl logs <pod-name> --previous
```

For the multicontainer case, use the following command:

```
Bash
```

```
kubectl logs <pod-name> -c <container-name> --previous
```

5. Check whether there are any network policies that might block the traffic:

```
Bash
```

```
kubectl get networkpolicies -A
```

You should see output that resembles the following table.

```
Output
```

NAMESPACE	NAME	POD-SELECTOR	AGE
kube-system	konnectivity-agent	app=konnectivity-agent	4d1h

If you see any other network policy that's custom-created, check whether that policy is blocking access to or from the pods.

6. Check whether you can reach the application from the service IP address. First, show details about the service resource, such as the external IP address and port, by running the `kubectl get services` command:

Bash

```
kubectl get services -n <namespace-name>
```

Output

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
my-service	LoadBalancer	10.0.21.43	20.119.121.232	80:31773/TCP 28s

Then, run cURL by using the service IP address and port to check whether you can reach the application:

Console

```
curl -Iv http://20.119.121.232:80
.
.
.
< HTTP/1.1 200 OK
HTTP/1.1 200 OK
```

7. Get more verbose information about the service:

Bash

```
kubectl describe services <service-name> -n <namespace-name>
```

8. Check the pod's IP address:

Bash

```
kubectl get pods -o wide
```

Output

NAME	READY	STATUS	RESTARTS	AGE	IP
my-pod	1/1	Running	0	12m	10.244.0.15
aks-agentpool-0000000-vmss000000					

9. Verify that the pod's IP address exists as an endpoint in the service:

Bash

```
kubectl describe services my-cluster-ip-service
```

Output

```
Name:           my-cluster-ip-service
Namespace:      default
Selector:       app=my-pod
Type:          ClusterIP
IP Family Policy: SingleStack
IP Families:    IPv4
IP:            10.0.174.133
IPs:           10.0.174.133
Port:          <unset>  80/TCP
TargetPort:     80/TCP
Endpoints:     10.244.0.15:80    # <--- Here
```

10. Verify the endpoints directly:

Bash

```
kubectl get endpoints
```

Output

NAME	ENDPOINTS	AGE
my-cluster-ip-service	10.244.0.15:80	14m

11. If the connection to a service doesn't work, restart the `kube-proxy` and CoreDNS pods:

Bash

```
kubectl delete pods -n kube-system -l component=kube-proxy
kubectl delete pods -n kube-system -l k8s-app=kube-dns
```

12. Verify that the node isn't overused:

Bash

```
kubectl top nodes
```

ⓘ Note

You can also use [Azure Monitor](#) to get the usage data for the cluster.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Troubleshoot connections to endpoints in the same virtual network

Article • 10/23/2024

This article discusses how to troubleshoot connections to endpoints in the same virtual network from a Microsoft Azure Kubernetes Service (AKS) cluster.

Troubleshooting checklist

The troubleshooting checklist covers connections to the following items:

- Virtual machines (VMs) or endpoints in the same subnet or a different subnet
- VMs or endpoints in a peered virtual network
- Private endpoints

Step 1: Do basic troubleshooting

Make sure that you can connect to endpoints. For instructions, see [Basic troubleshooting of outbound AKS cluster connections](#).

Step 2: Check for private endpoints

If the connection goes through a private endpoint, follow the instructions in [Troubleshoot Azure Private Endpoint connectivity problems](#).

Step 3: Check for virtual network peering

If the connection uses virtual network peering, follow the instructions in [Troubleshoot a connectivity issue between two peered virtual networks](#).

Step 4: Check whether the AKS network security group blocks traffic in the outbound scenario

To use AKS to check the network security groups (NSGs) and their associated rules, follow these steps:

1. In the [Azure portal](#), search for and select **Virtual machine scale sets**.

2. In the list of scale set instances, select the instance that you're using.

3. In the menu pane of your scale set instance, select **Networking**.

The **Networking** page for the scale set instance appears. In the **Outbound port rules** tab, two sets of rules are displayed. These rule sets are based on the two NSGs that act on the scale set instance. The following table describes the rule sets.

 Expand table

NSG rule level	NSG rule name	Attached to	Notes
Subnet	< <i>my-aks-nsg</i> >	The < <i>my-aks-subnet</i> > subnet	This is a common arrangement if the AKS cluster uses a custom virtual network and a custom subnet.
Network adapter	aks-agentpool-< <i>agentpool-number</i> >-nsg	The aks-agentpool-< <i>vm-scale-set-number</i> >-vmss network interface	This NSG is applied by the AKS cluster and managed by AKS.

By default, the egress traffic is allowed on the NSGs. However, in some scenarios, the custom NSG that's associated with the AKS subnet might have a **Deny** rule that has a more urgent priority. This rule stops traffic to some endpoints.

AKS doesn't modify the egress rules in an NSG. If AKS uses a custom NSG, verify that it allows egress traffic for the endpoints on the correct port and protocol. To make sure that the AKS cluster functions as expected, verify that the egress for custom NSGs allows [Required outbound network rules for AKS clusters](#).

Custom NSGs can directly affect the egress traffic by blocking the outbound rules. They can also affect egress traffic indirectly. For more information, see [A custom network security group blocks traffic](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Provide product feedback 

Troubleshoot connections to endpoints outside the virtual network

Article • 10/18/2024

This article discusses how to troubleshoot connections to endpoints outside the virtual network (that is, through the public internet) from a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- [Azure CLI](#).
- The client URL ([curl ↗](#)) tool.
- The Kubernetes [kubectl ↗](#) tool, or a similar tool to connect to the cluster. To install kubectl by using Azure CLI, run the [az aks install-cli](#) command.

Troubleshooting checklist

Is the issue persistent?

Step 1: Do basic troubleshooting

Make sure that you can connect to public endpoints on the internet. For instructions, see [Basic troubleshooting of outbound AKS cluster connections](#).

Step 2: Determine the outbound type for the AKS cluster

To identify the outbound type of the AKS cluster, run the [az aks show](#) command:

Azure CLI

```
az aks show --resource-group <resource_group> --name <cluster_name> --query "networkProfile.outboundType"
```

If the outbound type is `loadBalancer`, there's no route table unless you use *kubenet* network. If you use *kubenet*, make sure that the default route table has no additional configuration that blocks outbound internet connection. If you use another network, such as Azure CNI, Dynamic Allocation, or Azure CNI Overlay, no route table is created

by default. In this case, make sure the NSG (network security group) has no custom configuration that blocks outbound internet connection.

If the outbound type is `userDefinedRouting`, make sure that the following conditions are met:

- The egress device (firewall or proxy) is reachable.
- The egress device allows the [required outbound traffic](#) from the cluster.

To get the list of FQDNs that are allowed for your AKS cluster, run the [az aks egress-endpoints list](#) command:

Azure CLI

```
az aks egress-endpoints list --resource-group <resource_group> --name <cluster_name>
```

If the outbound type is `managedNATGateway`, check whether the AKS subnet is associated with the NAT gateway by running the [az network nat gateway show](#) command:

Azure CLI

```
az network nat gateway show --resource-group <resource_group> --name <nat_gateway_name> --query "subnets[].id"
```

For more information about how to use a NAT gateway together with AKS, see [Managed NAT gateway](#).

Step 3: Examine the curl output when you connect to the application pod

The curl response codes can help you identify the issue type. After the response code becomes available, try to better understand how the issue behaves. For more information about the HTTP status codes and the underlying behavior of the issue, refer to the following table.

[+] [Expand table](#)

Information source	Link
Internet Assigned Numbers Authority (IANA)	Hypertext Transfer Protocol (HTTP) status code registry

Information source	Link
Mozilla	HTTP response status codes ↗
Wikipedia	List of HTTP status codes ↗

The following HTTP status codes might indicate the listed issues.

[] [Expand table](#)

HTTP status code	Issue	Example
4xx	<ul style="list-style-type: none"> 1. An issue affects the client request. 2. A network blocker exists between the client and the server. 	<ul style="list-style-type: none"> 1. The requested page doesn't exist, or the client doesn't have permission to access the page. 2. Traffic is being blocked by a network security group or a firewall.
5xx	An issue affects the server.	The application is down, or a gateway isn't working.

You can try to connect to the application endpoint using curl. Here's an example command and output:

```
Bash

# 404 error code example
$ curl -vv <host IP address>/test.index

*   Trying <host IP address>:80...
* TCP_NODELAY set
* Connected to <host IP address> (<host IP address>) port 80 (#0)
> GET /test.index HTTP/1.1
> Host: <host IP address>
> User-Agent: curl/7.68.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 404 Not Found
...
* Connection #0 to host <host IP address> left intact
```

Step 4: Check what happens if outbound traffic bypasses the virtual appliance temporarily

For quick testing to determine whether the egress device (virtual appliance) causes the issue, you can temporarily allow all traffic to go through the internet. To configure this setup, you can change the default IP address and port route of `0.0.0.0/0` through the virtual appliance to go through the internet instead.

If the egress device works well when bypassing the virtual appliance, check the logs of the virtual appliance to see which packets are denied and add allow rules in the virtual appliance accordingly.

Is the issue intermittent?

You may experience intermittent outbound issues for many reasons. For troubleshooting intermittent outbound connection issues, try the following steps:

Step 1: Check if the pod or node resources are exhausted

Run the following commands to check how much the resources are used:

```
Bash
```

```
kubectl top pods  
kubectl top nodes
```

Step 2: Check if the operating system disk is used heavily

To check whether the operating system disk is used heavily, follow these steps:

1. In the [Azure portal](#), search for and select **Virtual machine scale sets**.
2. In the list of scale sets, select the scale set that's used for your AKS cluster.
3. In the scale set navigation pane, go to the **Monitoring** section, and then select **Metrics**.
4. View the disk metrics for the scale set from the **Metrics** section by looking for the following fields.

 Expand table

Field	Value
Scope	VMSS Name

Field	Value
Metrics Namespace	Virtual Machine Host
Metrics	OS and Data Disk Metric

For more information about metrics, see [OS Disk and Data Disk metrics](#).

To view AKS recommendations about disk utilization, follow these steps:

1. In the [Azure portal](#), search for and select **Kubernetes services**.
2. In the list of Kubernetes services, select the name of your AKS cluster.
3. In the AKS cluster navigation pane, go to the **Monitoring** section, and then select **Advisor recommendations**.
4. Review the listed recommendations about disk usage.

If the OS disk is used heavily, consider using the following remedies:

- Increase the OS disk size.
- Switch to [Ephemeral OS disks](#).

If these remedies don't resolve the issue, analyze the process that does heavy read/write operations on the disk. Then, check whether you can move the actions to a data disk instead of the OS disk.

Step 3: Check if the source network address translation port is exhausted

If applications are making many outbound connections, they may exhaust the number of available ports on the outbound device's IP address. Follow [Standard load balancer diagnostics with metrics, alerts, and resource health](#) to monitor the usage and allocation of your existing load balancer's [source network address translation \(SNAT\) port](#). Monitor to verify or determine the risk of [SNAT port exhaustion](#).

Are you reaching or exceeding the maximum number of allocated SNAT ports? In that case, you can check your application to determine whether it's reusing existing connections. For more information, see [Design your applications to use connections efficiently](#).

If you feel that the application is configured correctly, and you do need more SNAT ports than the default number of allocated ports, follow these steps:

1. Increase the number of public IPs on the egress device. If the egress device is the load balancer, you can [Increase the number of public IP addresses on the load balancer](#).
2. [Increase the ports per node for your AKS worker nodes](#).

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?



Traffic between node pools is blocked by a custom network security group

Article • 09/23/2024

This article discusses how to resolve a scenario in which a custom network security group (NSG) blocks traffic between node pools in a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

The Domain Name System (DNS) resolution from pods of the User node pool fails.

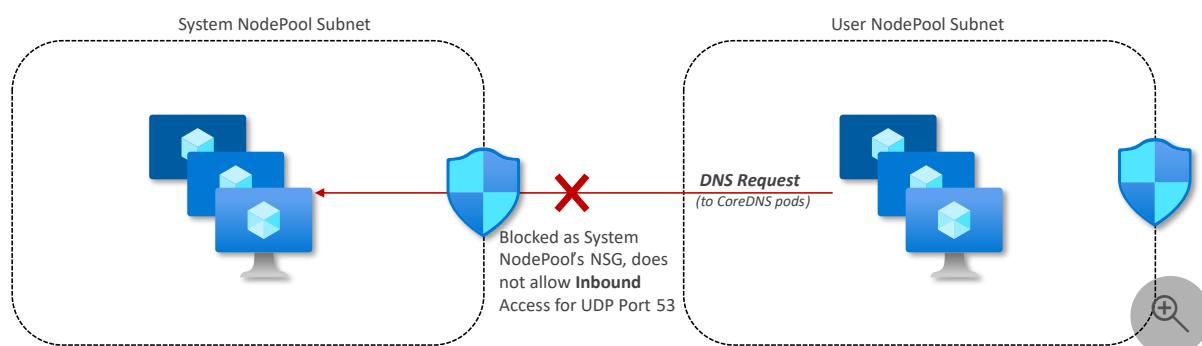
Background

In scenarios that involve multiple node pools, the pods in the `kube-system` namespace can be put on one node pool (the System node pool) while the application pods are put on a different node pool (the User node pool). In some scenarios, pods that communicate with one another can be in different node pools.

AKS lets customers have [node pools in different subnets](#). This feature means that customers can also associate different NSGs with each node pool's subnet.

Cause 1: The NSG of a node pool blocks inbound traffic

Inbound access on the NSG of a node pool blocks traffic. For example, a custom NSG on the System node pool (that hosts the core DNS pods) blocks inbound traffic on User Datagram Protocol (UDP) port 53 from the subnet of the User node pool.

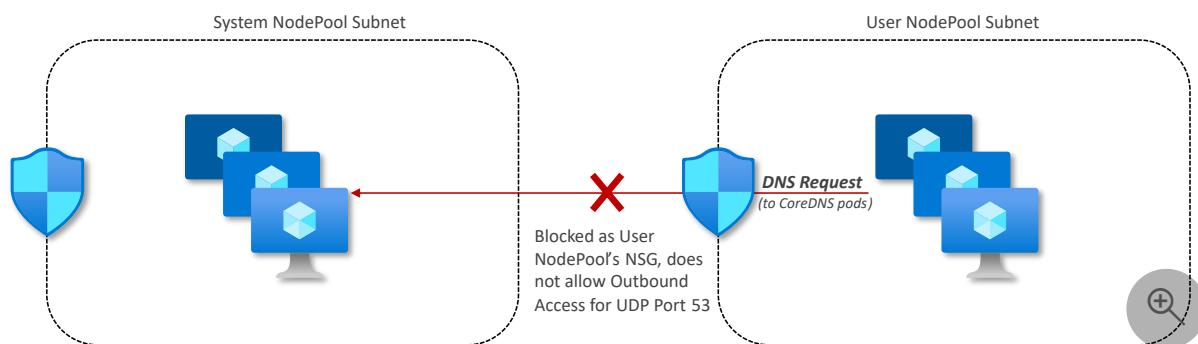


Solution 1: Configure the custom NSG to allow traffic between the node pools

Make sure that your custom NSG allows the required traffic between the node pools, specifically on UDP port 53. AKS won't update the custom NSG that's associated with subnets.

Cause 2: Outbound access on the NSG of a node pool blocks traffic

The NSG on another node pool blocks outbound access to the pod. For example, a custom NSG on the User node pool blocks outbound traffic on UDP port 53 to the System node pool.



Solution 2: Configure the custom NSG to allow traffic between the node pools

See [Solution 1: Configure the custom NSG to allow traffic between the node pools](#).

Cause 3: TCP port 10250 is blocked

Another common scenario is that Transmission Control Protocol (TCP) port 10250 is blocked between the node pools.

In this situation, a user might receive an error message that resembles the following text:

Console

```
$ kubectl logs nginx-57cdfd6dd9-xb7hk
Error from server: Get https://<node>:10250/containerLogs/default/nginx-
```

```
57cdfd6dd9-xb7hk/nginx: net/http: TLS handshake timeout
```

If the communication on TCP port 10250 is blocked, the connection to the Kubelet will be affected, and the logs won't be fetched.

Solution 3: Check for NSG rules that block TCP port 10250

Check whether any NSG rules block node-to-node communication on TCP port 10250.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?



Can't view resources in Kubernetes resource viewer in Azure portal

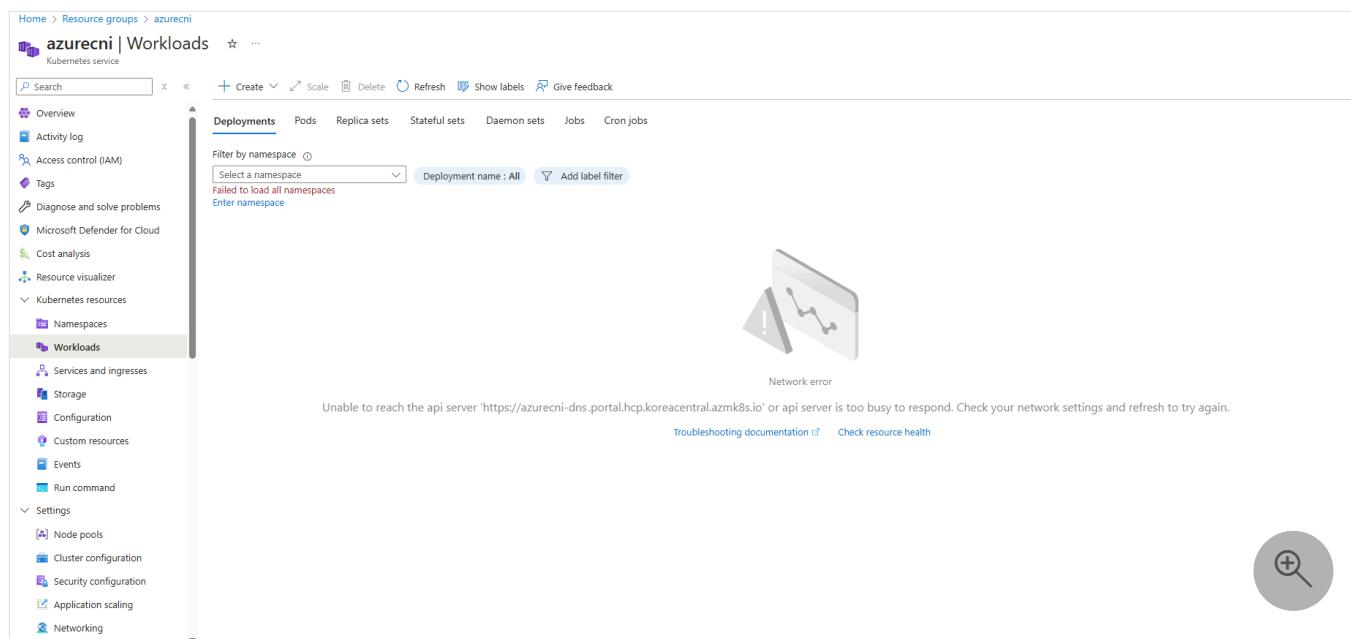
Article • 04/14/2025

This article discusses how to resolve an issue that prevents you from using the Azure portal to view resources in the Kubernetes resource viewer.

Symptoms

When you try to view resources in the [Kubernetes resource viewer](#) in the [Azure portal](#), you receive the following error message:

```
Unable to reach the API server '<server-url>' or API is too busy to response. Check your network setting.
```



Cause 1: You configured authorized IP ranges

You configured your Microsoft Azure Kubernetes Service (AKS) cluster to access the cluster API server by using authorized IP address ranges that your computer can't access.

Solution

When you run the `az aks create` or `az aks update` command in [Azure CLI](#), make sure that the `--api-server-authorized-ip-ranges` parameter includes access for the local client computer to the IP addresses or IP address ranges from which the portal is being browsed.

Cause 2: AKS is a private cluster

Your AKS is created as a private cluster, and you're accessing the Azure portal from a network that can't communicate with the subnet to which your AKS is connected.

Solution

This behavior is expected. To view the AKS resources in the Azure portal, you must access the Azure portal from a client that's located on a network that has connectivity to the AKS subnet. Then, you can view all AKS-related resources in the Azure portal.

More information

- [How to find my IP to include in --api-server-authorized-ip-ranges?](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

"Error from server: error dialing backend: dial tcp" message

Article • 03/05/2025

Symptoms

You receive an "Error from server: error dialing backend: dial tcp" error message when you take one of the following actions:

- Use any of the `kubectl logs`, `exec`, `attach`, `top`, or `port-forward` commands.
- Use third-party Kubernetes client tools to achieve the same functionality as the commands in the previous list item.

Why the error occurs

The Kubernetes API server has to forward API requests to an upstream component for several use cases. This error occurs if the API server can't establish a TCP connection to the upstream component. Examples of such upstream components include Kubernetes services that are inside the cluster and kubelet.

If the issue persists, a network blockage is likely the cause. To identify the responsible network configuration, first determine the scope of the problem.

Narrowing down: Are all `kubectl` sub-commands failing?

Try to run at least the `kubectl exec`, `kubectl logs <podname>`, and `kubectl top pods` commands.

If only `kubectl logs <podname>` or `kubectl exec` fails, check whether the issue occurs by having pods on different nodes.

If only `kubectl top pods` fails, check whether the issue occurs for pods on all nodes or only for pods on one node.

Cause 1: kubelet port (node:10250) is blocked

Pod-specific access issues, such as those that are experienced by running `kubectl logs` and `kubectl exec`, occur if the API server cannot reach the node on port 10250 to

access the Kubelet API. These issues can be caused by a connection that's blocked by a Network Security Group (NSG) or firewall.

To resolve the issue, check whether the NSG on the node subnet includes an inbound rule that might block TCP port 10250.

Cause 2: Specific service failing

Kubernetes accesses `svc/metrics-server` under the `kube-system` namespace for running `kubectl top` commands. There are other scenarios, such as [admission webhooks](#), in which the API server can also reach other services. It is important to note that, depending on the service failure pattern, the error message may vary.

To troubleshoot the issue, check the error message to identify which service is affected and review the status of the related pods, services, and endpoints.

Cause 3: Konnectivity or tunnel failing

When [API Server VNet Integration](#) is not enabled, AKS deploys a tunnel solution that proxies API server requests to in-cluster networking locations. Most AKS clusters use the Konnectivity solution. Konnectivity does not require that you open special ports on the API server. For more information, see [AKS required network rules](#).

To resolve the issue, check whether the `konnectivity-agent` in the `kube-system` namespace is running and its containers are in a ready state. Try to delete the pods to see whether the connection recovers after the new pods are ready.

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Provide product feedback 

InsufficientSubnetSize error code

Article • 02/20/2025

This article discusses how to resolve an `InsufficientSubnetSize` error that occurs when you try to deploy a Microsoft Azure Kubernetes Service (AKS) cluster that uses advanced networking. This article applies to both Kubernetes clusters and Azure Container Networking Interface (CNI) clusters.

Symptoms

The `InsufficientSubnetSize` error occurs during any of the following operations. This error is also encountered in [AKS Diagnostics](#), which proactively discovers issues such as an insufficient subnet size.

Operation 1: Scaling an AKS cluster or an AKS node pool

[+] Expand table

Cluster type	Symptom: The number of free IP addresses in the subnet is less than...
Kubenet	The number of new nodes that are requested.
Azure CNI	The number of new nodes that are requested times the node pool value in the <code>--max-pod</code> parameter.
Azure CNI Overlay	The number of new nodes that are requested. (In the node pools that use the autoscaler, the number of nodes is the value in the <code>--max-count</code> parameter.)

Operation 2: Upgrading an AKS cluster or an AKS node pool

[+] Expand table

Cluster type	Symptom: The number of free IP addresses in the subnet is less than...
Kubenet	The number of buffer nodes that have to be upgraded.
Azure CNI	The number of buffer nodes that have to be upgraded times the node pool value in the <code>--max-pod</code> parameter.

Cluster type	Symptom: The number of free IP addresses in the subnet is <i>less than...</i>
Azure CNI	The number of buffer nodes that have to be upgraded.
Overlay	(In the node pools that use autoscaler, the number of nodes is the value in the <code>--max-count</code> parameter.)

By default, an AKS cluster sets a maximum surge (upgrade buffer) value of one (1). However, you can customize this upgrade behavior by setting the maximum surge value of a node pool. This action increases the number of available IP addresses that are necessary to complete an upgrade.

Operation 3: Creating an AKS cluster or adding an AKS node pool

[] Expand table

Cluster type	Symptom: The number of free IP addresses in the subnet is <i>less than...</i>
Kubenet	The number of nodes that are requested.
Azure CNI	The number of nodes that are requested times the node pool value in the <code>--max-pod</code> parameter.
Overlay	The number of nodes that are requested. (In the node pools that use the autoscaler, the number of nodes is the value in the <code>--max-count</code> parameter.)

Cause

A subnet that's in use for a cluster no longer has available IP addresses within its Classless Inter-Domain Routing (CIDR) address space for successful resource assignment.

[] Expand table

Cluster type	Requirement
Kubenet	Sufficient IP space for each <i>node</i> in the cluster
Azure CNI	Sufficient IP space for each <i>node and pod</i> in the cluster
Azure CNI Overlay	Sufficient IP space for each <i>node</i> in the cluster

Read more about the [design of Azure CNI to assign IP addresses to pods](#).

Solution

Trying to update a subnet's CIDR address space in an existing node pool isn't currently supported. To migrate your workloads to a new node pool in a larger subnet, follow these steps:

1. Create a subnet in the cluster virtual network that contains a larger CIDR address range than that of the existing subnet. For information about how to size the subnet adequately for your cluster, see [Plan IP addressing for your cluster](#).
2. Create a node pool on the new subnet by running the `az aks nodepool add` command together with the `--vnet-subnet-id` parameter.
3. Migrate your workloads to the new node pool by draining the nodes in the old node pool. For information about how to drain AKS worker nodes safely, see [Safely Drain a Node](#).
4. Delete the original node pool by running the `az aks nodepool delete` command.

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

10250 I/O timeouts error when running kubectl log command

07/09/2025

TCP timeouts can be caused by blockages of internal traffic that runs between nodes. To investigate TCP time-outs, verify that this traffic isn't being blocked, for example, by [network security groups](#) (NSGs) on the subnet for your cluster nodes.

Connect to the cluster

First, connect to your Azure Kubernetes Service (AKS) cluster by running the following command:

Bash

```
export RESOURCE_GROUP=<your-resource-group>
export CLUSTER_NAME=<your-cluster-name>

az aks get-credentials --resource-group $RESOURCE_GROUP --name $CLUSTER_NAME
```

Symptoms

Tunnel functionalities, such as `kubectl logs` and code execution, work only for pods that are hosted on nodes on which tunnel service pods are deployed. Pods on other nodes that have no tunnel service pods cannot reach to the tunnel. When viewing the logs of these pods, you receive the following error message:

Bash

```
kubectl logs $POD_NAME
```

Results:

Output

```
Error from server: Get "https://aks-agentpool-xxxxxxxxx-
xxxxxxxxxx:10250/containerLogs/vsm-mba-prod/mba-api-app-xxxxxxxxxx/technosvc":
dial tcp <IP-Address>:10250: i/o timeout
```

Solution

To resolve this issue, allow traffic on port 10250 as described in this [article](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

TCP time-outs when kubectl or other third-party tools connect to the API server

07/09/2025

This article discusses how to troubleshoot TCP time-outs that occur when [kubectl](#) or other third-party tools are used to connect to the API server in Microsoft Azure Kubernetes Service (AKS). To ensure its service-level objectives (SLOs) and service-level agreements (SLAs), AKS uses high-availability (HA) control planes that scale vertically and horizontally, based on the number of cores.

Symptoms

You experience repeated connection time-outs.

Cause 1: Pods that are responsible for node-to-control plane communication aren't running

If only a few of your API commands are timing out consistently, the following pods might not be in a running state:

- `konnectivity-agent`
- `tunnelfront`
- `aks-link`

Note

In newer AKS versions, `tunnelfront` and `aks-link` are replaced with `konnectivity-agent`, so you'll only see `konnectivity-agent`.

These pods are responsible for communication between a node and the control plane.

Solution: Reduce the utilization or stress of the node hosts

Make sure the nodes that host these pods aren't overly utilized or under stress. Consider moving the nodes to their own [system node pool](#).

To check which node the `konnectivity-agent` pod is hosted on and the usage of the node, run the following commands:

Set access to the AKS cluster. Replace the values of `ResourceGroupName` and `AKSClusterName` with your own.

Bash

```
az aks get-credentials --resource-group ${ResourceGroupName} --name ${AKSClusterName} --overwrite-existing
```

Check the running pods in the kube-system namespace and which node each one is assigned to:

Bash

```
kubectl get pod -n kube-system -o wide
```

Results:

Output

NAME	READY	STATUS	RESTARTS	AGE	IP
NODE		NOMINATED NODE		READINESS GATES	
konnectivity-agent-xxxxx	1/1	Running	0	22h	
10.xxx.xx.xxx aks-nodepool1-xxxxx-vmss00000			<none>		<none>
coredns-xxxxx	1/1	Running	0	22h	
10.xxx.xx.xxx aks-nodepool1-xxxxx-vmss00001			<none>		<none>
# ...other pods...					

Check the usage of the nodes and see resource utilization for each node:

Bash

```
kubectl top node
```

Results:

Output

NAME	CPU(cores)	CPU%	MEMORY(bytes)	MEMORY%
aks-nodepool1-xxxxx-vmss00000	125m	12%	1510Mi	37%
aks-nodepool1-xxxxx-vmss00001	106m	10%	1203Mi	42%
# ...other nodes...				

Cause 2: Access is blocked on some required ports, FQDNs, and IP addresses

If the required ports, fully qualified domain names (FQDNs), and IP addresses aren't all opened, several command calls might fail. Secure, tunneled communication on AKS between the API server and the [kubelet](#) (through the `konnectivity-agent` pod) requires some of those items to work successfully.

Solution: Open the necessary ports, FQDNs, and IP addresses

For more information about what ports, FQDNs, and IP addresses need to be opened, see [Outbound network and FQDN rules for Azure Kubernetes Service \(AKS\) clusters](#).

Cause 3: The Application-Layer Protocol Negotiation TLS extension is blocked

To establish a connection between the control plane and nodes, the `konnectivity-agent` pod requires the [Transport Layer Security \(TLS\) extension for Application-Layer Protocol Negotiation \(ALPN\)](#). You might have previously blocked this extension.

Solution: Enable the ALPN extension

Enable the ALPN extension on the `konnectivity-agent` pod to prevent TCP time-outs.

Cause 4: The API server's IP authorized ranges doesn't cover your current IP address

If you use authorized IP address ranges on your API server, your API calls will be blocked if your IP isn't included in the authorized ranges.

Solution: Modify the authorized IP address ranges so that it covers your IP address

Change the authorized IP address ranges so that your IP address is covered. For more information, see [Update a cluster's API server authorized IP ranges](#).

Cause 5: A client or application leaks calls to the API server

Frequent GET calls can accumulate and overload the API server.

Solution: Use watches instead of GET calls, but make sure the application doesn't leak those calls

Make sure that you use watches instead of frequent GET calls to the API server. You also have to make sure that your third-party applications don't leak any watch connections or GET calls. For example, in the [Istio microservice architecture](#), a [bug in the mixer application](#) creates a new API server watch connection whenever a secret is read internally. Because this behavior happens at a regular interval, the watch connections quickly accumulate. These connections eventually cause the API server to become overloaded no matter the scaling pattern.

Cause 6: Too many releases in your Helm deployments

If you use too many releases in your deployments of [Helm](#) (the Kubernetes package manager), the nodes start to consume too much memory. It also results in a large amount of [ConfigMap](#) (configuration data) objects, which might cause unnecessary usage spikes on the API server.

Solution: Limit the maximum number of revisions for each release

Because the maximum number of revisions for each release is infinite by default, you need to run a command to set this maximum number to a reasonable value. For Helm 2, the command is [helm init](#). For Helm 3, the command is [helm upgrade](#). Set the `--history-max <value>` parameter when you run the command.

 Expand table

Version	Command
Helm 2	<code>helm init --history-max <maximum-number-of-revisions-per-release> ...</code>
Helm 3	<code>helm upgrade ... --history-max <maximum-number-of-revisions-per-release> ...</code>

Cause 7: Internal traffic between nodes is being blocked

There might be internal traffic blockages between nodes in your AKS cluster.

Solution: Troubleshoot the "dial tcp <Node_IP>:10250: i/o timeout" error

See [Troubleshoot TCP timeouts, such as "dial tcp <Node_IP>:10250: i/o timeout"](#).

Cause 8: Your cluster is private

Your cluster is a private cluster, but the client from which you're trying to access the API server is in a public or different network that can't connect to the subnet used by AKS.

Solution: Use a client that can access the AKS subnet

Since your cluster is private and its control plane is in the AKS subnet, it can't be connected to the API server unless it's in a network that can connect to the AKS subnet. It's an expected behavior.

In this case, try to access the API server from a client in a network that can communicate with the AKS subnet. Additionally, verify network security groups (NSGs) or other appliances between networks aren't blocking packets.

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Troubleshoot DNS resolution problems in AKS

Article • 05/30/2025

This article discusses how to create a troubleshooting workflow to fix Domain Name System (DNS) resolution problems in Microsoft Azure Kubernetes Service (AKS).

Prerequisites

- The Kubernetes [kubectl](#) command-line tool

Note: To install kubectl by using [Azure CLI](#), run the `az aks install-cli` command.

- The [dig](#) command-line tool for DNS lookup
- The [grep](#) command-line tool for text search
- The [Wireshark](#) network packet analyzer

Troubleshooting checklist

Troubleshooting DNS problems in AKS is typically a complex process. You can easily get lost in the many different steps without ever seeing a clear path forward. To help make the process simpler and more effective, use the "scientific" method to organize the work:

- Step 1. Collect the facts.
- Step 2. Develop a hypothesis.
- Step 3. Create and implement an action plan.
- Step 4. Observe the results and draw conclusions.
- Step 5. Repeat as necessary.

Troubleshooting Step 1: Collect the facts

To better understand the context of the problem, gather facts about the specific DNS problem. By using the following baseline questions as a starting point, you can describe the nature of the problem, recognize the symptoms, and identify the scope of the problem.

[Expand table](#)

Question	Possible answers
Where does the DNS resolution fail?	<ul style="list-style-type: none">• Pod• Node• Both pods and nodes
What kind of DNS error do you get?	<ul style="list-style-type: none">• Time-out• No such host• Other DNS error
How often do the DNS errors occur?	<ul style="list-style-type: none">• Always• Intermittently• In a specific pattern
Which records are affected?	<ul style="list-style-type: none">• A specific domain• Any domain
Do any custom DNS configurations exist?	<ul style="list-style-type: none">• Custom DNS server configured on the virtual network• Custom DNS on CoreDNS configuration
What kind of performance problems are affecting the nodes?	<ul style="list-style-type: none">• CPU• Memory• I/O throttling

Question	Possible answers
What kind of performance problems are affecting the CoreDNS pods?	<ul style="list-style-type: none"> CPU Memory I/O throttling
What causes DNS latency?	DNS queries that take too much time to receive a response (more than five seconds)

To get better answers to these questions, follow this three-part process.

Part 1: Generate tests at different levels that reproduce the problem

The DNS resolution process for pods on AKS includes many layers. Review these layers to isolate the problem. The following layers are typical:

- CoreDNS pods
- CoreDNS service
- Nodes
- Virtual network DNS

To start the process, run tests from a test pod against each layer.

Test the DNS resolution at CoreDNS pod level

1. Deploy a test pod to run DNS test queries by running the following command:

```
Bash

cat <<EOF | kubectl apply --filename -
apiVersion: v1
kind: Pod
metadata:
  name: aks-test
spec:
  containers:
    - name: aks-test
      image: debian:stable
      command: ["/bin/sh"]
      args: ["-c", "apt-get update && apt-get install -y dnsutils && while true; do sleep 1000; done"]
EOF
```

2. Retrieve the IP addresses of the CoreDNS pods by running the following [kubectl get](#) command:

```
Bash

kubectl get pod --namespace kube-system --selector k8s-app=kube-dns --output wide
```

3. Connect to the test pod using the `kubectl exec -it aks-test -- bash` command and test the DNS resolution against each CoreDNS pod IP address by running the following commands:

```
Bash

# Placeholder values
FQDN=<fully-qualified-domain-name> # For example, "db.contoso.com"
DNS_SERVER=<coredns-pod-ip-address>

# Test loop
for i in $(seq 1 1 10)
do
    echo "host= $(dig +short ${FQDN} @${DNS_SERVER})"
    sleep 1
done
```

For more information about troubleshooting DNS resolution problems from the pod level, see [Troubleshoot DNS resolution failures from inside the pod](#).

Test the DNS resolution at CoreDNS service level

1. Retrieve the CoreDNS service IP address by running the following `kubectl get` command:

Bash

```
kubectl get service kube-dns --namespace kube-system
```

2. On the test pod, run the following commands against the CoreDNS service IP address:

Bash

```
# Placeholder values
FQDN=<fully-qualified-domain-name> # For example, "db.contoso.com"
DNS_SERVER=<kubedns-service-ip-address>

# Test loop
for i in $(seq 1 1 10)
do
    echo "host= $(dig +short ${FQDN} @${DNS_SERVER})"
    sleep 1
done
```

Test the DNS resolution at node level

1. Connect to the node.
2. Run the following `grep` command to retrieve the list of upstream DNS servers that are configured:

Bash

```
grep ^nameserver /etc/resolv.conf
```

3. Run the following text commands against each DNS that's configured in the node:

Bash

```
# Placeholder values
FQDN=<fully-qualified-domain-name> # For example, "db.contoso.com"
DNS_SERVER=<dns-server-in-node-configuration>

# Test loop
for i in $(seq 1 1 10)
do
    echo "host= $(dig +short ${FQDN} @${DNS_SERVER})"
    sleep 1
done
```

Test the DNS resolution at virtual network DNS level

Examine the DNS server configuration of the virtual network, and determine whether the servers can resolve the record in question.

Part 2: Review the health and performance of CoreDNS pods and nodes

Review the health and performance of CoreDNS pods

You can use `kubectl` commands to check the health and performance of CoreDNS pods. To do so, follow these steps:

1. Verify that the CoreDNS pods are running:

Bash

```
kubectl get pods -l k8s-app=kube-dns -n kube-system
```

2. Check if the CoreDNS pods are overused:

Bash

```
kubectl top pods -n kube-system -l k8s-app=kube-dns
```

Output

NAME	CPU(cores)	MEMORY(bytes)
coredns-dc97c5f55-424f7	3m	23Mi
coredns-dc97c5f55-wbh4q	3m	25Mi

3. Get the nodes that host the CoreDNS pods:

Bash

```
kubectl get pods -n kube-system -l k8s-app=kube-dns -o jsonpath='{.items[*].spec.nodeName}'
```

4. Verify that the nodes aren't overused:

Bash

```
kubectl top nodes
```

5. Verify the logs for the CoreDNS pods:

Bash

```
kubectl logs -l k8s-app=kube-dns -n kube-system
```

ⓘ Note

To get more debugging information, enable verbose logs in CoreDNS. To do so, see [Troubleshooting CoreDNS customization in AKS](#).

Review the health and performance of nodes

You might first notice DNS resolution performance problems as intermittent errors, such as time-outs. The main causes of this problem include resource exhaustion and I/O throttling within nodes that host the CoreDNS pods or the client pod.

To check whether resource exhaustion or I/O throttling is occurring, run the following `kubectl describe` command together with the `grep` command on your nodes. This series of commands lets you review the request count and compare it against the limit for each resource. If the limit percentage is more than 100 percent for a resource, that resource is overcommitted.

Bash

```
kubectl describe node | grep -A5 '^Name:\|^\$Allocated resources:' | grep -v '.kubernetes.io\|^\$Roles:\|^\$Labels:'
```

The following snippet shows example output from this command:

Output

```
Name:           aks-nodepool1-17046773-vmss00000m
--
Allocated resources:
  (Total limits might be more than 100 percent.)
  Resource      Requests      Limits
  -----        -----        -----
  cpu          250m (13 percent)  40m (2 percent)
  memory       420Mi (9 percent)  1762Mi (41 percent)
--
Name:           aks-nodepool1-17046773-vmss00000n
--
Allocated resources:
  (Total limits might be more than 100 percent.)
  Resource      Requests      Limits
  -----        -----        -----
  cpu          612m (32 percent)  8532m (449 percent)
  memory       804Mi (18 percent)  6044Mi (140 percent)
--
Name:           aks-nodepool1-17046773-vmss00000o
--
Allocated resources:
  (Total limits might be more than 100 percent.)
  Resource      Requests      Limits
  -----        -----        -----
  cpu          250m (13 percent)  40m (2 percent)
  memory       420Mi (9 percent)  1762Mi (41 percent)
```

```
--  
Name: aks-ubuntu-16984727-vmss000008  
--  
Allocated resources:  
(Total limits might be more than 100 percent.)  
Resource Requests Limits  
----- -----  
cpu 250m (13 percent) 40m (2 percent)  
memory 420Mi (19 percent) 1762Mi (82 percent)
```

To get a better picture of resource usage at the pod and node level, you can also use Container insights and other cloud-native tools in Azure. For more information, see [Monitor Kubernetes clusters using Azure services and cloud native tools](#).

Part 3: Analyze DNS traffic and review DNS resolution performance

Analyzing DNS traffic can help you understand how your AKS cluster handles the DNS queries. Ideally, you should reproduce the problem on a test pod while you capture the traffic from this test pod and on each of the CoreDNS pods.

There are two main ways to analyze DNS traffic:

- Using real-time DNS analysis tools, such as [Inspektor Gadget](#), to analyze the DNS traffic in real time.
- Using traffic capture tools, such as [Retina Capture](#) and [Dumpy](#), to collect the DNS traffic and analyze it with a network packet analyzer tool, such as Wireshark.

Both approaches aim to understand the health and performance of DNS responses using DNS response codes, response times, and other metrics. Choose the one that fits your needs best.

Real-time DNS traffic analysis

You can use [Inspektor Gadget](#) to analyze the DNS traffic in real time. To install Inspektor Gadget to your cluster, see [How to install Inspektor Gadget in an AKS cluster](#).

To trace DNS traffic across all namespaces, use the following command:

```
Bash  
  
# Get the version of Gadget  
GADGET_VERSION=$(kubectl gadget version | grep Server | awk '{print $3}')  
# Run the trace_dns gadget  
kubectl gadget run trace_dns:$GADGET_VERSION --all-namespaces --fields "src,dst,name,qr,qtype,id,rcode,latency_ns"
```

Where `--fields` is a comma-separated list of fields to be displayed. The following list describes the fields that are used in the command:

- `src`: The source of the request with Kubernetes information (`<kind>/<namespace>/<name>:<port>`).
- `dst`: The destination of the request with Kubernetes information (`<kind>/<namespace>/<name>:<port>`).
- `name`: The name of the DNS request.
- `qr`: The query/response flag.
- `qtype`: The type of the DNS request.
- `id`: The ID of the DNS request, which is used to match the request and response.
- `rcode`: The response code of the DNS request.
- `latency_ns`: The latency of the DNS request.

The command output looks like the following:

Output					
SRC RCODE	LATENCY_NS	DST	NAME	QR QTYPE	ID
p/default/aks-test:33141 0ns		p/kube-system/coredns-57d886c994-r2...	db.contoso.com.	Q A	c215
p/kube-system/coredns-57d886c994-r2... 0ns	168.63.129.16:53		db.contoso.com.	Q A	323c
168.63.129.16:53 NameErr...	13.64ms	p/kube-system/coredns-57d886c994-r2...	db.contoso.com.	R A	323c
NameErr... p/default/aks-test:56921 0ns	0ns	p/kube-system/coredns-57d886c994-r2...	db.contoso.com.	R A	c215
			db.contoso.com.	Q A	6574

```
p/kube-system/coredns-57d886c994-r2... p/default/aks-test:56921  
NameErr... 0ns
```

db.contoso.com.

R A

6574

You can use the `ID` field to identify whether a query has a response. The `RCODE` field shows you the response code of the DNS request. The `LATENCY_NS` field shows you the latency of the DNS request in nanoseconds. These fields can help you understand the health and performance of DNS responses. For more information about real-time DNS analysis, see [Troubleshoot DNS failures across an AKS cluster in real time](#).

Capture DNS traffic

This section demonstrates how to use Dumpy to collect DNS traffic captures from each CoreDNS pod and a client DNS pod (in this case, the `aks-test` pod).

To collect the captures from the test client pod, run the following command:

Bash

```
kubectl dumpy capture pod aks-test -f "-i any port 53" --name dns-cap1-aks-test
```

To collect captures for the CoreDNS pods, run the following Dumpy command:

Bash

```
kubectl dumpy capture deploy coredns \  
-n kube-system \  
-f "-i any port 53" \  
--name dns-cap1-coredns
```

Ideally, you should be running captures while the problem reproduces. This requirement means that different captures might be running for different amounts of time, depending on how often you can reproduce the problem. To collect the captures, run the following commands:

Bash

```
mkdir dns-captures  
kubectl dumpy export dns-cap1-aks-test ./dns-captures  
kubectl dumpy export dns-cap1-coredns ./dns-captures -n kube-system
```

To delete the Dumpy pods, run the following Dumpy command:

Bash

```
kubectl dumpy delete dns-cap1-coredns -n kube-system  
kubectl dumpy delete dns-cap1-aks-test
```

To merge all the CoreDNS pod captures, use the `mergecap` command line tool for merging capture files. The `mergecap` tool is included in the Wireshark network packet analyzer tool. Run the following `mergecap` command:

Bash

```
mergecap -w coredns-cap1.pcap dns-cap1-coredns-<coredns-pod-name-1>.pcap dns-cap1-coredns-<coredns-pod-name-2>.pcap [...]
```

DNS packet analysis for an individual CoreDNS pod

After you generate and merge your traffic capture files, you can do a DNS packet analysis of the capture files in Wireshark. Follow these steps to view the packet analysis for the traffic of an individual CoreDNS pod:

1. Select **Start**, enter **Wireshark**, and then select **Wireshark** in the search results.
2. In the **Wireshark** window, select the **File** menu, and then select **Open**.
3. Navigate to the folder that contains your capture files, select `dns-cap1-db-check-<db-check-pod-name>.pcap` (the client-side capture file for an individual CoreDNS pod), and then select the **Open** button.
4. Select the **Statistics** menu, and then select **DNS**. The **Wireshark - DNS** dialog box appears and displays an analysis of the DNS traffic. The contents of the dialog box are shown in the following table.

[Expand table](#)

Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
▼ Total Packets	1066				0.0017	100%	0.1200	0.000
▼ rcode	1066				0.0017	100.00%	0.1200	0.000
Server failure	17				0.0000	1.59%	0.0100	99.332
No such name	353				0.0006	33.11%	0.0400	0.000
No error	696				0.0011	65.29%	0.0800	0.000
▼ opcodes	1066				0.0017	100.00%	0.1200	0.000
Standard query	1066				0.0017	100.00%	0.1200	0.000
▼ Query/Response	1066				0.0017	100.00%	0.1200	0.000
Response	531				0.0009	49.81%	0.0600	0.000
Query	535				0.0009	50.19%	0.0600	0.000
▼ Query Type	1066				0.0017	100.00%	0.1200	0.000
AAAA	167				0.0003	15.67%	0.0200	0.000
A	899				0.0015	84.33%	0.1000	0.000
▼ Class	1066				0.0017	100.00%	0.1200	0.000
IN	1066				0.0017	100.00%	0.1200	0.000
▼ Service Stats	0				0.0000	100%	-	-
request-response time (msec)	531	184.42	0.067000	6308.503906	0.0009		0.0600	0.000
no. of unsolicited responses	0				0.0000		-	-
no. of retransmissions	0				0.0000		-	-
▼ Response Stats	0				0.0000	100%	-	-
no. of questions	1062	1.00	1	1	0.0017		0.1200	0.000
no. of authorities	1062	0.82	0	1	0.0017		0.1200	0.000
no. of answers	1062	0.15	0	1	0.0017		0.1200	0.000
no. of additionals	1062	0.00	0	0	0.0017		0.1200	0.000
▼ Query Stats	0				0.0000	100%	-	-
Qname Len	535	32.99	14	66	0.0009		0.0600	0.000
▼ Label Stats	0				0.0000		-	-
4th Level or more	365				0.0006		0.0400	0.000
3rd Level	170				0.0003		0.0200	0.000
2nd Level	0				0.0000		-	-
1st Level	0				0.0000		-	-
Payload size	1066	92.87	32	194	0.0017	100%	0.1200	0.000

The DNS analysis dialog box in Wireshark shows a total of 1,066 packets. Of these packets, 17 (1.59 percent) caused a server failure. Additionally, the maximum response time was 6,308 milliseconds (6.3 seconds), and no response was received for 0.38 percent of the queries. (This total was calculated by subtracting the 49.81 percent of packets that contained responses from the 50.19 percent of packets that contained queries.)

If you enter `(dns.flags.response == 0) && ! dns.response_in` as a display filter in Wireshark, this filter displays DNS queries that didn't receive a response, as shown in the following table.

[Expand table](#)

No.	Time	Source	Destination	Protocol	Length	Info
225	2024-04-01 16:50:40.000520	10.0.0.21	172.16.0.10	DNS	80	Standard query 0x2c67 AAAA db.contoso.com
426	2024-04-01 16:52:47.419907	10.0.0.21	172.16.0.10	DNS	132	Standard query 0x8038 A db.contoso.com.iosffyoulcwehgo1g3egb3m4oc.jx.internal.cloudapp.net
693	2024-04-01 16:55:23.105558	10.0.0.21	172.16.0.10	DNS	132	Standard query 0xbcb0 A db.contoso.com.iosffyoulcwehgo1g3egb3m4oc.jx.internal.cloudapp.net
768	2024-04-01 16:56:06.512464	10.0.0.21	172.16.0.10	DNS	80	Standard query 0xe330 A db.contoso.com

Additionally, the Wireshark status bar displays the text **Packets: 1066 - Displayed: 4 (0.4%)**. This information means that four of the 1,066 packets, or 0.4 percent, were DNS queries that never received a response. This percentage essentially matches the 0.38 percent total that you calculated earlier.

If you change the display filter to `dns.time >= 5`, the filter shows query response packets that took five seconds or more to be received, as shown in the updated table.

[Expand table](#)

No.	Time	Source	Destination	Protocol	Length	Info	SourcePort	Addit...
213	2024-04-01 16:50:32.644592	172.16.0.10	10.0.0.21	DNS	132	Standard query 0x9312 Server failure A db.contoso.com.iosffyoulcwehgo1g3egb3m4oc.jx.internal.cloudapp.net	53	
320	2024-04-01 16:51:55.053896	172.16.0.10	10.0.0.21	DNS	80	Standard query 0xe5ce Server failure A db.contoso.com	53	
328	2024-04-01 16:51:55.113619	172.16.0.10	10.0.0.21	DNS	132	Standard query 0x6681 Server failure A db.contoso.com.iosffyoulcwehgo1g3egb3m4oc.jx.internal.cloudapp.net	53	
335	2024-04-01 16:52:02.553811	172.16.0.10	10.0.0.21	DNS	80	Standard query 0x6cf6 Server failure A db.contoso.com	53	
541	2024-04-01 16:53:53.423838	172.16.0.10	10.0.0.21	DNS	80	Standard query 0x07b3 Server failure AAAA db.contoso.com	53	
553	2024-04-01 16:54:05.165234	172.16.0.10	10.0.0.21	DNS	132	Standard query 0x1ea0 Server failure A db.contoso.com.iosffyoulcwehgo1g3egb3m4oc.jx.internal.cloudapp.net	53	
774	2024-04-01 16:56:17.553531	172.16.0.10	10.0.0.21	DNS	80	Standard query 0xa20f Server failure AAAA db.contoso.com	53	
891	2024-04-01 16:56:44.442334	172.16.0.10	10.0.0.21	DNS	132	Standard query 0xa279 Server failure A db.contoso.com.iosffyoulcwehgo1g3egb3m4oc.jx.internal.cloudapp.net	53	

After you change the display filter, the status bar is updated to show the text **Packets: 1066 - Displayed: 8 (0.8%)**. Therefore, eight of the 1,066 packets, or 0.8 percent, were DNS responses that took five seconds or more to be received. However, on most clients, the default DNS time-out value is expected to be five seconds. This expectation means that, although the CoreDNS pods processed and delivered the eight responses, the client already ended the session by issuing a "timed out" error message. The **Info** column in the filtered results shows that all eight packets caused a server failure.

DNS packet analysis for all CoreDNS pods

In Wireshark, open the capture file of the CoreDNS pods that you merged earlier (`coredns-cap1.pcap`), and then open the DNS analysis, as described in the previous section. A Wireshark dialog box appears that displays the following table.

[Expand table](#)

Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
▼ Total Packets	4540233				7.3387	100%	84.7800	592.950
▼ rcode	4540233				7.3387	100.00%	84.7800	592.950
Server failure	121781				0.1968	2.68%	8.4600	599.143
No such name	574658				0.9289	12.66%	10.9800	592.950

Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
No error	3843794				6.2130	84.66%	73.2500	592.950
▼ opcodes	4540233				7.3387	100.00%	84.7800	592.950
Standard query	4540233				7.3387	100.00%	84.7800	592.950
▼ Query/Response	4540233				7.3387	100.00%	84.7800	592.950
Response	2135116				3.4512	47.03%	39.0400	581.680
Query	2405117				3.8876	52.97%	49.1400	592.950
▼ Query Type	4540233				7.3387	100.00%	84.7800	592.950
SRV	3647				0.0059	0.08%	0.1800	586.638
PTR	554630				0.8965	12.22%	11.5400	592.950
NS	15918				0.0257	0.35%	0.7200	308.225
MX	393016				0.6353	8.66%	7.9700	426.930
AAAA	384032				0.6207	8.46%	8.4700	438.155
A	3188990				5.1546	70.24%	57.9600	592.950
▼ Class	4540233				7.3387	100.00%	84.7800	592.950
IN	4540233				7.3387	100.00%	84.7800	592.950
▼ Service Stats	0				0.0000	100%	-	-
request-response time (msec)	2109677	277.18	0.020000	12000.532227	3.4100		38.0100	581.680
no. of unsolicited responses	25402				0.0411		5.1400	587.832
no. of retransmissions	37				0.0001		0.0300	275.702
▼ Response Stats	0				0.0000	100%	-	-
no. of questions	4244830	1.00	1	1	6.8612		77.0500	581.680
no. of authorities	4244830	0.39	0	11	6.8612		77.0500	581.680
no. of answers	4244830	1.60	0	22	6.8612		77.0500	581.680
no. of additionals	4244830	0.29	0	26	6.8612		77.0500	581.680
▼ Query Stats	0				0.0000	100%	-	-
Qname Len	2405117	20.42	2	113	3.8876		49.1400	592.950
▼ Label Stats	0				0.0000		-	-
4th Level or more	836034				1.3513		16.1900	592.950
3rd Level	1159513				1.8742		23.8900	592.950
2nd Level	374182				0.6048		8.7800	592.955
1st Level	35388				0.0572		0.9200	294.492
Payload size	4540233	89.87	17	1128	7.3387	100%	84.7800	592.950

The dialog box indicates that there were a combined total of about 4.5 million (4,540,233) packets, of which 2.68 percent caused server failure. The difference in query and response packet percentages shows that 5.94 percent of the queries (52.97 percent minus 47.03 percent) didn't receive a response. The maximum response time was 12 seconds (12,000.532227 milliseconds).

If you apply the display filter for query responses that took five seconds or more (`dns.time >= 5`), most of the packets in the filter results will indicate that a server failure occurred. This is probably because of a client "timed out" error.

The following table is a summary of the capture findings.

[Expand table](#)

Capture review criteria	Yes	No
Difference between DNS queries and responses exceeds two percent	<input checked="" type="checkbox"/>	<input type="checkbox"/>
DNS latency is more than one second	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Troubleshooting Step 2: Develop a hypothesis

This section categorizes common problem types to help you narrow down potential problems and identify components that might require adjustments. This approach sets the foundation for creating a targeted action plan to mitigate and resolve these problems effectively.

Common DNS response codes

The following table summarizes the most common DNS return codes.

[Expand table](#)

DNS return code	DNS return message	Description
RCODE:0	NOERROR	The DNS query finished successfully.
RCODE:1	FORMERR	A DNS query format error exists.
RCODE:2	SERVFAIL	The server didn't complete the DNS request.
RCODE:3	NXDOMAIN	The domain name doesn't exist.
RCODE:5	REFUSED	The server refused to answer the query.
RCODE:8	NOTAUTH	The server isn't authoritative for the zone.

General problem types

The following table lists problem type categories that help you break down the problem symptoms.

[Expand table](#)

Problem type	Description
Performance	DNS resolution performance problems can cause intermittent errors, such as "timed out" errors from a client's perspective. These problems might occur because nodes experience resource exhaustion or I/O throttling. Additionally, constraints on compute resources in CoreDNS pods can cause resolution latency. If CoreDNS latency is high or increases over time, this might indicate a load problem. If CoreDNS instances are overloaded, you might experience DNS name resolution problems and delays, or you might see problems in workloads and Kubernetes internal services.
Configuration	Configuration problems can cause incorrect DNS resolution. In this case, you might experience NXDOMAIN or "timed out" errors. Incorrect configurations might occur in CoreDNS, nodes, Kubernetes, routing, virtual network DNS, private DNS zones, firewalls, proxies, and so on.
Network connectivity	Network connectivity problems can affect pod-to-pod connectivity (east-west traffic) or pod-and-node connectivity to external resources (north-south traffic). This scenario can cause "timed out" errors. The connectivity problems might occur if the CoreDNS service endpoints aren't up to date (for example, because of kube-proxy problems, routing problems, packet loss, and so on). External resource dependency combined with connectivity problems (for example, dependency on custom DNS servers or external DNS servers) can also contribute to the problem.

Required inputs

Before you formulate a hypothesis of probable causes for the problem, summarize the results from the previous steps of the troubleshooting workflow.

You can collect the results by using the following tables.

Results of the baseline questionnaire template

[Expand table](#)

Question	Possible answers
Where does the DNS resolution fail?	<input type="checkbox"/> Pod <input type="checkbox"/> Node

Question	Possible answers
	<input type="checkbox"/> Both pod and node
What type of DNS error do you get?	<input type="checkbox"/> Timed out <input type="checkbox"/> NXDOMAIN <input type="checkbox"/> Other DNS error
How often do the DNS errors occur?	<input type="checkbox"/> Always <input type="checkbox"/> Intermittently <input type="checkbox"/> In a specific pattern
Which records are affected?	<input type="checkbox"/> A specific domain <input type="checkbox"/> Any domain
Do any custom DNS configurations exist?	<input type="checkbox"/> Custom DNS servers on a virtual network <input type="checkbox"/> Custom CoreDNS configuration

Results of tests at different levels

[Expand table](#)

Resolution test results	Works	Fails
From pod to CoreDNS service	<input type="checkbox"/>	<input type="checkbox"/>
From pod to CoreDNS pod IP address	<input type="checkbox"/>	<input type="checkbox"/>
From pod to Azure internal DNS	<input type="checkbox"/>	<input type="checkbox"/>
From pod to virtual network DNS	<input type="checkbox"/>	<input type="checkbox"/>
From node to Azure internal DNS	<input type="checkbox"/>	<input type="checkbox"/>
From node to virtual network DNS	<input type="checkbox"/>	<input type="checkbox"/>

Results of health and performance of the nodes and the CoreDNS pods

[Expand table](#)

Performance review results	Healthy	Unhealthy
Nodes performance	<input type="checkbox"/>	<input type="checkbox"/>
CoreDNS pods performance	<input type="checkbox"/>	<input type="checkbox"/>

Results of traffic captures and DNS resolution performance

[Expand table](#)

Capture review criteria	Yes	No
Difference between DNS queries and responses exceeds two percent	<input type="checkbox"/>	<input type="checkbox"/>
DNS latency is more than one second	<input type="checkbox"/>	<input type="checkbox"/>

Map required inputs to problem types

To develop your first hypothesis, map each of the results from the required inputs to one or more of the problem types. By analyzing these results in the context of problem types, you can develop hypotheses about the potential root causes of the DNS resolution problems. Then, you can create an action plan of targeted investigation and troubleshooting.

Error type mapping pointers

- If test results show DNS resolution failures at the CoreDNS service, or contain "timed out" errors when trying to reach specific endpoints, then configuration or connectivity problems might exist.
- Indications of compute resource starvation at CoreDNS pod or node levels might suggest performance problems.

- DNS captures that have a considerable mismatch between DNS queries and DNS responses can indicate that packets are being lost. This scenario suggests that there are connectivity or performance problems.
- The presence of custom configurations at the virtual network level or Kubernetes level can contain setups that don't work with AKS and CoreDNS as expected.

Troubleshooting Step 3: Create and implement an action plan

You should now have enough information to create and implement an action plan. The following sections contain extra recommendations to formulate your plan for specific problem types.

Performance problems

If you're dealing with DNS resolution performance problems, review and implement the following best practices and guidance.

 Expand table

Best practice	Guidance
Set up a dedicated system node pool that meets minimum sizing requirements.	Manage system node pools in Azure Kubernetes Service (AKS)
To avoid disk I/O throttling, use nodes that have Ephemeral OS disks.	Default OS disk sizing and GitHub issue 1373 in Azure AKS
Follow best resource management practices on workloads within the nodes.	Best practices for application developers to manage resources in Azure Kubernetes Service (AKS)

If DNS performance still isn't good enough after you make these changes, consider using [Node Local DNS](#).

Configuration problems

Depending on the component, you should review and understand the implications of the specific setup. See the following list of component-specific documentation for configuration details:

- [Kubernetes DNS configuration options](#)
- [AKS CoreDNS custom configuration options](#)
- [Private DNS zones missing a virtual network link](#)

Network connectivity problems

- Bugs that involve the Container Networking Interface (CNI) or other Kubernetes or OS components usually require intervention from AKS support or the AKS product group.
- Infrastructure problems, such as hardware failures or hypervisor problems, might require collaboration from infrastructure support teams. Alternatively, these problems might have self-healing features.

Troubleshooting Step 4: Observe results and draw conclusions

Observe the results of implementing your action plan. At this point, your action plan should be able to fix or mitigate the problem.

Troubleshooting Step 5: Repeat as necessary

If these troubleshooting steps don't resolve the problem, repeat the troubleshooting steps as necessary.

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Real-time DNS traffic analysis to troubleshoot DNS resolution failures in AKS

Article • 05/30/2025

This article discusses how to troubleshoot Domain Name System (DNS) failures across an Azure Kubernetes Service (AKS) cluster in real time.

! Note

This article is complementary to [Basic troubleshooting of DNS resolution problems in AKS](#) guide.

Symptoms

The following table outlines common symptoms that you might observe in an AKS cluster:

 Expand table

Symptom	Description
High error rate	DNS queries fail or return unexpected results, which can affect the performance and reliability of applications that depend on them.
Unresponsive services	DNS queries take longer than usual to resolve, which can cause delays or time-outs in applications that rely on them.
Service discovery is impacted	Applications can't locate other applications in the cluster due to DNS issues, which can lead to service disruptions or failures.
External communication is impacted	Applications have trouble accessing external resources or endpoints outside the cluster due to DNS problems, which can result in errors or degraded performance.

Prerequisites

- Azure CLI.

To install Azure CLI, see [How to install the Azure CLI](#).

- A tool to connect to the cluster, such as the Kubernetes command-line tool [kubectl](#) ↗.

To install kubectl by using [Azure CLI](#), run the `az aks install-cli` command.

- [Inspektor Gadget](#).

This tool is used in most troubleshooting steps covered in this article, so make sure to install it on the cluster. To install it on an AKS cluster, see [How to install Inspektor Gadget in an AKS cluster](#).

- Familiarity with [gadgets](#).
- [Inspektor Gadget DNS gadget](#).

It's used in all the following troubleshooting steps.

To troubleshoot DNS failures across an AKS cluster, use the instructions in the following sections. Before we start, export the gadget version to a variable. This variable is used in all the commands in this article.

Bash

```
GADGET_VERSION=$(kubectl gadget version | grep Server | awk '{print $3}')
```

Please feel free to choose a different version. For example, you can use `latest` or `main` to get the latest stable or development version of the gadget respectively.

Step 1: Identify unsuccessful DNS responses across the cluster

You can use the DNS gadget to identify all the unsuccessful DNS responses across the cluster. To perform this check, trace DNS packets on all the nodes and filter unsuccessful responses.

Bash

```
kubectl gadget run trace_dns:$GADGET_VERSION \
--all-namespaces \
--fields k8s.node,src,dst,name,qtype,rcode \
--filter "qr==R,rcode!=Success"
```

Here are the explanations of the command parameters:

- `--all-namespaces`: Shows data from pods in all namespaces.
- `--fields k8s.node,src,dst,name,qtype,rcode`: Shows only Kubernetes information, source and destination of the DNS packets, DNS name, DNS query type, and DNS response code.

- `--filter "qr==R,rcode!=Success"`: Matches only DNS responses (`qr==R`) with a response code other than `Success`.

The output should look similar to the following:

Output

K8S.NODE NAME	SRC QTYPE	RCODE	DST
aks-agentpool-...919-vmss000000	p/kube-system/coredns-57d886c994-r2ts...		
p/default/mypod:38480		myaks-troubleshoot.	A
NameError			
aks-agentpool-...919-vmss000000	s/kube-system/kube-dns:53		
p/default/mypod:38480		myaks-troubleshoot.	A
NameError			

Where `RCODE` is the DNS response code and helps you identify the type of DNS failure.

Here are some causes of unsuccessful DNS responses:

- The DNS name being resolved has a typo.
- The upstream DNS nameservers experience issues.
- A misconfigured networking configuration (such as NetworkPolicy (NetPol), firewall or NSG rules) blocks DNS traffic.
- The DNS name becomes invalid after expansion.

To understand how DNS queries might be expanded using the `/etc/resolv.conf` file of a pod, see [Namespaces of Services ↗](#).

Step 2: Identify slow DNS queries across the cluster

You can use the DNS gadget to identify all the slow DNS queries across the cluster. To perform this check, trace DNS packets on all the nodes and filter slow responses.

In the following example, you're using a latency value of `5ms` to define slow packets. You can change it to a desired value, for example, `5μs`, `20ms`, `1s`:

Bash

```
kubectl gadget run trace_dns:$GADGET_VERSION \
--all-namespaces \
--fields k8s.node,src,dst,name,qtype,rcode,latency_ns \
--filter "latency_ns_raw>5000000"
```

Here are the explanations of the command parameters:

- `--all-namespaces` : Shows data from pods in all namespaces.
- `--fields k8s.node,src,dst,name,qtype,rcode,latency_ns` : Shows only Kubernetes information, source and destination of the DNS packets, DNS name, DNS query type, DNS response code, and latency in nanoseconds.
- `--filter "latency_ns_raw>5000000"` : Matches only DNS responses with latency greater than `5ms` (`5000000` nanoseconds).

The output should look similar to the following:

Output				
K8S.NODE	SRC	QTYPE	RCODE	DST
NAME				LATENCY_NS
aks-agentpool...9-vmss000001	168.63.129.16:53			p/kube-
system/coredns-57d886c994...	microsoft.com.		A	Success
14.02ms				
aks-agentpool...9-vmss000000	s/kube-system/kube-dns:53			
p/default/mypod:39168	microsoft.com.		A	Success
15.40ms				
aks-agentpool...9-vmss000000	168.63.129.16:53			p/kube-
system/coredns-57d886c994...	microsoft.com.		AAAA	Success
5.90ms				
aks-agentpool...9-vmss000000	s/kube-system/kube-dns:53			
p/default/mypod:38953	microsoft.com.		AAAA	Success
6.41ms				

Here are some causes of slow DNS queries:

- The upstream DNS nameservers experience issues.
- Networking issues in the cluster.

Step 3: Verify the health of the upstream DNS servers

You can use the DNS gadget to verify the health of the upstream DNS servers used by CoreDNS. If applications try to reach external domains, the queries are forwarded to the upstream DNS servers via CoreDNS. To understand the health of these queries, trace the DNS packets leaving the CoreDNS pods and filter by the nameserver.

In the following example, the [default Azure DNS Server](#) (IP address 168.63.129.16) is used as the upstream nameserver. If you're using a custom DNS server, you can use the IP address of the custom DNS server as the upstream nameserver. You can get the IP address by looking up `/etc/resolv.conf` on the node.

Bash

```
kubectl gadget run trace_dns:$GADGET_VERSION \
--all-namespaces \
--fields src,dst,id,qr,name,nameserver,rcode,latency_ns \
--filter "nameserver.addr==168.63.129.16"
```

Here are the explanations of the command parameters:

- `--all-namespaces`: Shows data from pods in all namespaces.
- `--fields src,dst,id,qr,name,nameserver,rcode,latency_ns`: Shows only source and destination of the DNS packets, DNS query ID, query/response, DNS name, nameserver, DNS response result, and latency in nanoseconds.
- `--filter "nameserver.addr==168.63.129.16"` : Matches only DNS packets with the nameserver IP address `168.63.129.16`.

The output should look similar to the following:

Output

SRC	DST		
ID	QR NAME	NAMESERVER	RCODE
LATENCY_NS			
p/kube-system/coredns-57d886c994-vs... 4828 0ns	r/168.63.129.16:53 Q qasim-cluster-dns-sqia0j5i.hc...	168.63.129.16	
p/kube-system/coredns-57d886c994-pcv59:51... 5015 0ns	r/168.63.129.16:53 Q qasim-cluster-dns-sqia0j5i.hc...	168.63.129.16	
r/168.63.129.16:53 pcv59:51... 5015 Success	R qasim-cluster-dns-sqia0j5i.hc... 5.06ms	168.63.129.16	p/kube-system/coredns-57d886c994-
r/168.63.129.16:53 vsj7g:60... 4828 Success	R qasim-cluster-dns-sqia0j5i.hc... 23.01ms	168.63.129.16	p/kube-system/coredns-57d886c994-

You can use the `RCODE`, and `LATENCY` values to determine the health of the upstream DNS server. For example, if there's an unhealthy upstream server, you see the following output:

- A DNS query (`QR=Q`) with an ID (for example, `4828`) has no matching response.
- A DNS response (`QR=R`) has a high value under the `LATENCY_NS` column (for example, `23.01ms`).
- A DNS response (`QR=R`) has an `RCODE` other than `Success`, for example, "ServerFailure" and "Refused".

Step 4: Verify DNS queries get responses in a timely manner

You can use the DNS gadget to verify that a particular DNS query gets a response in a timely manner. To perform this check, filter the events with a DNS name and match the query/response IDs:

Bash

```
kubectl gadget run trace_dns:$GADGET_VERSION \
  -l app=test-pod \
  --fields k8s.node,k8s.namespace,k8s.podname,id,qtype,qr,name,rcode,latency_ns
 \
  --filter name==microsoft.com.
```

Here are the explanations of the command parameters:

- `-l app=test-pod`: Shows only data to/from pods with the label `app=test-pod` in the default namespace.
- `--fields k8s.node,k8s.namespace,k8s.podname,id,qtype,qr,name,rcode,latency_ns`: Shows only Kubernetes information, DNS query ID, query type, query/response, DNS name, DNS response result, and latency in nanoseconds.
- `--filter name==microsoft.com.`: Matches only DNS packets with the DNS name `microsoft.com.`. Ensure that the filter value is a fully qualified domain name (**FQDN**) by adding a dot (.) to the end of the name.

The output should look similar to the following:

Output

K8S.NODE	K8S.NAMESPACE	RCODE	K8S.PODNAME	ID
QTYPE	QR NAME		LATENCY_NS	
aks-agentpoo...9-vmss000000	default		mypod	102d
A	Q microsoft.com.			0ns
aks-agentpoo...9-vmss000000	default		mypod	102d
A	R microsoft.com.	Success		11.87ms
aks-agentpoo...9-vmss000000	default		mypod	d482
AAAA	Q microsoft.com.			0ns
aks-agentpoo...9-vmss000000	default		mypod	d482
AAAA	R microsoft.com.	Success		9.27ms

The `ID` value (for example, `102d`) can be used to correlate the queries with responses. You also can use the `LATENCY_NS` value to validate that you get the responses in a timely manner.

Step 5: Verify DNS responses contain the expected IP addresses

You can use the DNS gadget to verify that a particular DNS query gets the expected response. For example, for a [headless service](#) (named `myheadless`), you would expect the response to contain the IP addresses of all the pods.

Bash

```
kubectl gadget run trace_dns:$GADGET_VERSION \
--fields k8s.podname,id,qtype,qr,name,rcode,num_answers,addresses \
--filter name~myheadless
```

Here are the explanations of the command parameters:

- `--fields k8s.podname,id,qtype,qr,name,rcode,num_answers,addresses`: Shows only Kubernetes information, DNS query ID, query type, query/response, DNS name, DNS response result, number of answers, and IP addresses in the response.
- `--filter name~myheadless`: Matches only DNS packets with the DNS name that contains `myheadless`. The `~` operator is used to match a regex.

The output should look similar to the following:

Output

K8S.PODNAME	ID	QTYPE	QR
NAME	RCODE	NUM_ANSWERS	ADDRESSES
mypod	f8b0	0	A
myheadless.default.svc.cluster.loc...			Q
mypod	f8b0	2	A
myheadless.default.svc.cluster.loc...	Success	2	10.244.0.146,10.24...
mypod	abcd		AAAA
myheadless.default.svc.cluster.loc...		0	Q
mypod	abcd		AAAA
myheadless.default.svc.cluster.loc...	Success	0	R

You can use the `NUM_ANSWERS` and `ADDRESSES` values to match the values you get from `kubectl get ep myheadless`.

Bash

```
kubectl get ep myheadless
NAME           ENDPOINTS          AGE
myheadless    10.244.0.146:80,10.244.1.207:80  10d
```

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Troubleshoot DNS resolution failures from inside the pod but not from the worker node

Article • 05/30/2025

This article discusses how to troubleshoot Domain Name System (DNS) resolution failures that occur from inside the pod, but not from the worker node, when you try to establish an outbound connection from a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- The Kubernetes [kubectl](#) tool, or a similar tool to connect to the cluster. To install kubectl by using [Azure CLI](#), run the `az aks install-cli` command.
- The [apt-get](#) command-line tool for handling packages.
- The [host](#) command-line tool for DNS lookups.
- The [systemctl](#) command-line tool.

Background

For DNS resolution, the pods send requests to the CoreDNS pods in the `kube-system` namespace.

If the DNS query is for an internal component, such as a service name, the CoreDNS pod responds by itself. However, if the request is for an external domain, the CoreDNS pod sends the request to the upstream DNS server.

The upstream DNS servers are obtained based on the `resolv.conf` file of the worker node in which the pod is running. The `resolv.conf` file (`/run/systemd/resolve/resolv.conf`) is updated based on the DNS settings of the virtual network on which the worker node is running.

Troubleshooting checklist

To troubleshoot DNS issues from within the pod, use the instructions in the following sections.

Step 1: Troubleshoot DNS issues from within the pod

You can use kubectl commands to troubleshoot DNS issues from within the pod, as shown in the following steps:

1. Verify that the CoreDNS pods are running:

```
Bash
```

```
kubectl get pods -l k8s-app=kube-dns -n kube-system
```

2. Check whether the CoreDNS pods are overused:

```
Console
```

```
$ kubectl top pods -n kube-system -l k8s-app=kube-dns
NAME           CPU(cores)   MEMORY(bytes)
coredns-dc97c5f55-424f7   3m          23Mi
coredns-dc97c5f55-wbh4q   3m          25Mi
```

3. Verify that the nodes that host the CoreDNS pods aren't overused. Also, get the nodes that are hosting the CoreDNS pods:

```
Bash
```

```
kubectl get pods -n kube-system -l k8s-app=kube-dns -o
jsonpath='{{.items[*].spec.nodeName}'
```

4. Check the usage of these nodes:

```
Bash
```

```
kubectl top nodes
```

5. Verify the logs for the CoreDNS pods:

```
Bash
```

```
kubectl logs -l k8s-app=kube-dns -n kube-system
```

(!) Note

To see more debugging information, enable verbose logs in CoreDNS. To enable verbose logging in CoreDNS, see [Troubleshooting CoreDNS customizations in AKS](#).

Step 2: Create a test pod to run commands

If DNS resolution is failing, follow these steps:

1. Run a test pod in the same namespace as the problematic pod [↗](#).

2. Start a test pod in the cluster:

Bash

```
kubectl run -it --rm aks-ssh --namespace <namespace> --image=debian:stable
```

When the test pod is running, you'll gain access to the pod.

3. Run the following commands to install the required packages:

Bash

```
apt-get update -y  
apt-get install dnsutils -y
```

4. Verify that the *resolv.conf* file has the correct entries:

Console

```
cat /etc/resolv.conf  
search default.svc.cluster.local svc.cluster.local cluster.local  
00idcnmrrm4edot5s2or1onxsc.bx.internal.cloudapp.net  
nameserver 10.0.0.10  
options ndots:5
```

5. Use the `host` command to determine whether the DNS requests are being routed to the upstream server:

Console

```
$ host -a microsoft.com  
Trying "microsoft.com.default.svc.cluster.local"  
Trying "microsoft.com.svc.cluster.local"  
Trying "microsoft.com.cluster.local"  
Trying "microsoft.com.00idcnmrrm4edot5s2or1onxsc.bx.internal.cloudapp.net"  
Trying "microsoft.com"  
Trying "microsoft.com"  
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62884  
;; flags: qr rd ra; QUERY: 1, ANSWER: 27, AUTHORITY: 0, ADDITIONAL: 5  
  
;; QUESTION SECTION:  
;microsoft.com. IN ANY
```

```
;; ANSWER SECTION:  
microsoft.com.          30      IN      NS      ns1-39.azure-dns.com.  
...  
...  
ns4-39.azure-dns.info.  30      IN      A       13.107.206.39  
  
Received 2121 bytes from 10.0.0.10#53 in 232 ms
```

6. Check the upstream DNS server from the pod to determine whether the DNS resolution is working correctly. For example, for Azure DNS, run the following [nslookup](#) command:

```
Console  
  
$ nslookup microsoft.com 168.63.129.16  
Server:      168.63.129.16  
Address:     168.63.129.16#53  
...  
...  
Address: 20.81.111.85
```

Step 3: Check whether DNS requests work when the upstream DNS server is explicitly specified

If the DNS requests from pods are working when you specify the upstream DNS server explicitly, verify the following conditions:

1. Check for a custom ConfigMap for CoreDNS:

```
Bash  
  
kubectl describe cm coredns-custom -n kube-system
```

If a custom ConfigMap is present, verify that the configuration is correct. For more information, see [Customize CoreDNS with Azure Kubernetes Service](#).

2. Check whether a network policy is blocking traffic on User Datagram Protocol (UDP) port 53 to the CoreDNS pods in the `kube-system` namespace.
3. Check whether the CoreDNS pods are on a different node pool (System node pool). If they are, check whether a network security group (NSG) is associated with the System node pool that's blocking traffic on UDP port 53.
4. Check whether the virtual network was updated recently to add the new DNS servers.

If a virtual network update occurred, check whether one of the following events has also occurred:

- The nodes were restarted.
- The network service in the node was restarted.

For the update in DNS settings to take effect, the network service on the node and the CoreDNS pods have to be restarted. To restart the network service or the pods, use one of the following methods:

- Restart the node.
- Scale new nodes. (New nodes will have the updated configuration.)
- Restart the network service in the nodes, and then restart the CoreDNS pods. Follow these steps:
 - a. Make a Secure Shell (SSH) connection to the nodes. For more information, see [Connect to Azure Kubernetes Service \(AKS\) cluster nodes for maintenance or troubleshooting](#).
 - b. From within the node, restart the network service:

```
Bash
```

```
systemctl restart systemd-networkd
```

- c. Check whether the settings are updated:

```
Bash
```

```
cat /run/systemd/resolve/resolv.conf
```

- d. After the network service is restarted, use kubectl to restart the CoreDNS pods:

```
Bash
```

```
kubectl delete pods -l k8s-app=kube-dns -n kube-system
```

5. Check whether more than one DNS server is specified in the virtual network DNS settings.

If multiple DNS servers are specified in the AKS virtual network, one of the following sequences occurs:

- The AKS node sends a request to the upstream DNS server as part of a series. In this sequence, the request is sent to the first DNS server that's configured in the virtual network (if the DNS servers are reachable and running). If the first DNS server isn't reachable and isn't responding, the request is sent to the next DNS server.

AKS nodes use the [resolver](#) command to send requests to the DNS servers. The configuration file for this resolver can be found at `/run/systemd/resolve/resolv.conf` in the AKS nodes.

Are there multiple servers? In this case, the resolver library queries them in the order that's listed. (The strategy used is to try a name server first. If the query times out, try the next name server, and continue until the list of name servers is exhausted. Then, the query continues to try to connect to the name servers until the maximum number of retries are made.)

- CoreDNS uses the [forward](#) plug-in to send requests to upstream DNS servers. This plug-in uses a random algorithm to select the upstream DNS server. In this sequence, the request could go to any of the DNS servers that are mentioned in the virtual network. For example, you might receive the following output:

Console

```
$ kubectl describe cm coredns -n kube-system
Name:         coredns
Namespace:    kube-system
Labels:       addonmanager.kubernetes.io/mode=Reconcile
              k8s-app=kube-dns
              kubernetes.io/cluster-service=true
Annotations: <none>

Data
=====
Corefile:
-----
.:53 {
    errors
    ready
    health
    kubernetes cluster.local in-addr.arpa ip6.arpa {
        pods insecure
        fallthrough in-addr.arpa ip6.arpa
    }
    prometheus :9153
    forward . /etc/resolv.conf                               # Here!
    cache 30
    loop
    reload
    loadbalance
    import custom/*.override
```

```
}
```

```
import custom/*.server
```



```
BinaryData
```

```
=====
```

```
Events: <none>
```

In the CoreDNS `forward` plugin, `policy` specifies the policy to use for selecting upstream servers. The policies are defined in the following table.

 [Expand table](#)

Policy name	Description
<code>random</code>	A policy that implements random upstream selection. This policy is the default policy.
<code>round_robin</code>	A policy that selects hosts based on round robin ordering.
<code>sequential</code>	A policy that selects hosts based on sequential ordering.

If the AKS virtual network contains multiple DNS servers, the requests from the AKS node might go to the first DNS server. However, the requests from the pod might go to second DNS server.

Cause: Multiple destinations for DNS requests

If two custom DNS servers are specified, and the third DNS server is specified as Azure DNS (168.63.129.16), the node will send requests to the first custom DNS server if it's running and reachable. In this setup, the node can resolve the custom domain. However, some of the DNS requests from the pod might be directed to Azure DNS. This is because CoreDNS can select the upstream server at random. In this scenario, the custom domain can't be resolved. Therefore, the DNS request fails.

Solution: Remove Azure DNS from virtual network settings

We recommend that you don't combine Azure DNS with custom DNS servers in the virtual network settings. If you want to use the custom DNS servers, add only the custom DNS servers

in the virtual network settings. Then, configure Azure DNS in the forwarder settings of your custom DNS servers.

For more information, see [Name resolution that uses your own DNS server](#).

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Capture real-time system insights from an AKS cluster

07/02/2025

This article discusses the process of gathering real-time system insights from your Microsoft Azure Kubernetes Service (AKS) cluster by using Inspektor Gadget. The article contains step-by-step instructions for installing this tool on your AKS environment. It also explores practical examples that show how Inspektor Gadget helps you gather valuable information to do effective debugging of real-world issues.

Demo: Real-time DNS troubleshooting and critical file-access alerting

To begin, consider the following quick demo. Suppose that you have to figure out why the DNS requests from an application fail. By using Inspektor Gadget, you can run the [trace_dns gadget](#) to capture the DNS traffic in the Kubernetes namespace in which your application is running:

Bash

```
kubectl gadget run trace_dns \
--namespace my-ns \
--fields k8s.node,k8s.podName,id,qr,name,rcode,nameserver
```

Output

K8S.NODE NAMESERVER	K8S.PODNAME	ID	QR	NAME	RCODE
aks-nodepool-41788306-vmss000002 1.2.3.4	demo-pod	13cc	Q	example.com.	
aks-nodepool-41788306-vmss000002 1.2.3.4	demo-pod	13cc	Q	example.com.	

From this information, we can see that the DNS requests are directed to the DNS server at IP address 1.2.3.4 (as indicated in the NAMESERVER column), but we only see the queries (Q in the QR column) and no responses (R in the QR column). This means that the DNS server didn't respond to the queries, which is why the application can't resolve the domain name www.example.com.

Now, suppose that 1.2.3.4 isn't the default name server configuration, and you suspect that a malicious process is modifying the configuration at runtime. In these kinds of cases, Inspektor

Gadget goes beyond DNS diagnostics. It also enables you to monitor processes that access critical files (such as `/etc/resolv.conf`) and have the intention of modifying those files. To do that, run the [trace_open gadget](#) in the same namespace and filter the results by the file name and the flags that indicate [the intention to write to the file](#) (`O_WRONLY` to open for writing only, or `O_RDWR` to open for reading and writing):

Bash

```
kubectl gadget run trace_open \
--namespace my-ns \
--filter fname==/etc/resolv.conf,flags~'(O_WRONLY|O_RDWR)' \
--fields k8s.node,k8s.podName,comm,fname,flags,error
```

Output

K8S.NODE	FLAGS	K8S.PODNAME	COMM	FNAME
ak5-nodepool-41788306-vmss000002	O_WRONLY	demo-pod	malicious-proc	/etc/resolv.conf

What is Inspektor Gadget?

[Inspektor Gadget](#) is a framework that makes it easy to monitor, troubleshoot, and secure workloads running on Linux and Kubernetes. It consists of tools (*Gadgets*) that leverage [eBPF](#) programs. Their primary goal is to gather low-level kernel data to provide insights into specific system scenarios. The Inspektor Gadget framework manages the association of the collected data by using high-level references, such as Kubernetes resources. This integration makes sure that a seamless connection exists between low-level insights and their corresponding high-level context. The integration streamlines the troubleshooting process and the collection of relevant information.

Gadgets

Inspektor Gadget provides a set of built-in tools that are designed to debug and observe common situations on a system. For example, by using such gadgets, you can trace the following events in your cluster:

- Process creation
- File access
- Network activity, such as TCP connections or DNS resolution

The gadgets present the information that they collected by using different mechanisms. For instance, some gadgets can inform you about the system status at specific times. Other gadgets can report every time a given event occurs, or they can provide periodic updates.

These are just a few examples. The [official documentation](#) provides detailed descriptions and examples of each gadget so that you can determine the most suitable gadget for your specific use case. However, if you find a use case that the existing gadgets don't currently cover, Inspektor Gadget allows you to run your own eBPF programs by using the [run command](#). Because the Inspektor Gadget framework handles the building, packaging, and deployment of your custom programs, it streamlines the process for your unique requirements. Also, it gathers high-level metadata to enrich the data that you collect in your program.

Use cases

To complement the demo that's presented at the beginning of this article, we compiled a list of issues and practical scenarios that show how Inspektor Gadget helps you tackle debugging challenges. The following examples showcase the potential of Inspektor Gadget. But the capabilities of this tool extend beyond these scenarios. This makes Inspektor Gadget an invaluable asset for navigating the complexities of Kubernetes debugging and observability.

 [Expand table](#)

Problem area	Symptoms	Troubleshooting
Disk-intensive applications	High memory or CPU usage, or inconsistent node readiness	An application might consistently engage in disk read/write operations, such as extensive logging. By using Inspektor Gadget, you can identify in real time which containers generate more block I/O . Or, more specifically, you can find the container that causes more reads and writes into a file .
"It's always DNS"	High application latency, timeouts, or poor end-user experience	By using Inspektor Gadget, you can trace all the DNS queries and responses in the cluster. In particular, Inspektor Gadget provides the following information that helps you to determine whether the DNS is affecting your application's performance: <ul style="list-style-type: none">• Query success• Whether the response contains an error• The name server that's used for the lookup• The query-response latency
File system access	Application misbehaves or can't function correctly	The application might be unable to access specific configurations, logs, or other vital files in the file system. In such scenarios, Inspektor Gadget enables you to trace

Problem area	Symptoms	Troubleshooting
		<p>all the opened files ↗ inside pods to diagnose access issues. Whenever your application tries to open a file, you can discover the following information:</p> <ul style="list-style-type: none"> • The flags that are used to open the file (for example, <code>O_RDONLY</code>, <code>O_WRONLY</code>, <code>O_RDWR</code> ↗, and so on) • Whether the file opening attempt succeeds • The returned error (if the file opening attempt fails) <p>For instance, if the attempt to open the file fails because of error 2 (ENOENT ↗), the application is probably trying to open a file that doesn't exist. This means that you might have a typo in the code, or the file is available in a different path.</p>
Remote code execution (RCE)	Unauthorized code execution such as cryptojacking ↗ that's evident in high CPU usage during application idle periods	When attackers try to make this kind of attack on a system, they usually have to run the code by using <code>bash</code> . Inspektor Gadget enables you to trace the creation of new processes ↗ , particularly processes that involve critical commands such as <code>bash</code> .

How to install Inspektor Gadget in an AKS cluster

One-Click Inspektor Gadget deployment

By selecting the following button, an AKS cluster will be automatically created, and Inspektor Gadget will be deployed in the cluster. After the deployment is finished, you can explore all the features of Inspektor Gadget in the provided shell environment.



Deploy to Azure [↗](#)

Install Inspektor Gadget by running the "kubectl gadget" plug-in

This section outlines the steps for installing Inspektor Gadget in your AKS cluster by running the `kubectl gadget` plug-in. The installation consists of two parts:

1. Installing the `kubectl gadget` plug-in on your workstation
2. Running the `kubectl gadget` plug-in to deploy Inspektor Gadget in the cluster

⚠ Warning

Many mechanisms are available to deploy and use Inspektor Gadget. Each of these mechanisms is tailored to specific use cases and requirements. You can use the `kubectl` gadget plug-in to apply several of these mechanisms, but not all of them. For instance, deploying Inspektor Gadget by using the `kubectl gadget` plug-in depends on the availability of the Kubernetes API server. If you can't depend on such a component because its availability might be occasionally compromised, we recommend that you avoid using the `kubectl gadget` deployment mechanism. For more information about this and other use cases, see the [Inspektor Gadget documentation](#).

Prerequisites

- The Kubernetes [kubectl](#) command-line tool. If you have [Azure CLI](#), you can run the `az aks install-cli` command to install `kubectl`.
- An AKS cluster. If you don't have an AKS cluster, [create one by using Azure CLI or by using the Azure portal](#).

Part 1: Install the `kubectl` gadget plug-in on your workstation

Use the instructions for your OS:

- Azure Linux 3.0
- Ubuntu 18.04 / 20.04 / 22.04

(!) Note

To install a specific release or compile it from the source, see [Install kubectl gadget](#) on GitHub.

Azure Linux 3.0

1. Add the Microsoft Cloud-Native repository to your system:

Bash

```
echo "[azurelinux-cloud-native]
name=Azure Linux Cloud Native 3.0
baseurl=https://packages.microsoft.com/azurelinux/3.0/prod/cloud-
```

```
native/$(uname -i)
gpgkey=file:///etc/pki/rpm-gpg/MICROSOFT-RPM-GPG-KEY
gpgcheck=1
repo_gpgcheck=1
enabled=1
skip_if_unavailable=True
sslverify=1
" > /etc/yum.repos.d/azurelinux-cloud-native.repo
```

2. Install the `kubectl gadget` plug-in:

Bash

```
tdnf install --refresh -y kubectl-gadget
```

Now, verify the installation by running the `version` command:

Bash

```
kubectl gadget version
```

The command output shows you the version of the client (`kubectl gadget` plug-in) and that it isn't installed yet on the server (the cluster):

Output

```
Client version: vX.Y.Z
Server version: not installed
```

Part 2: Deploy Inspektor Gadget in the cluster

The following command deploys the [DaemonSet](#) controller.

ⓘ Note

Several options are available to customize the deployment, as shown in the following list:

- Use a specific container image
- Deploy to specific nodes
- Deploy into a custom namespace

To learn about these options, see the [Installing in the cluster](#) section of the official documentation.

Bash

```
kubectl gadget deploy
```

Verify the installation by running the `version` command again:

Bash

```
kubectl gadget version
```

This time, the client and the server are both shown to be correctly installed:

Output

```
Client version: vX.Y.Z  
Server version: vX.Y.Z
```

When deploying Inspektor Gadget with the `kubectl gadget` plug-in available in the Microsoft Cloud-Native repository, the container image used for the DaemonSet is automatically pulled from the Microsoft Container Registry (MCR):

Bash

```
kubectl get daemonset gadget -n gadget -o  
jsonpath='{.spec.template.spec.containers[*].image}'
```

Output

```
mcr.microsoft.com/oss/v2/inspektor-gadget/inspektor-gadget:vX.Y.Z
```

ⓘ Note

If you use the [krew](#) package manager to install `kubectl` plug-ins and prefer to install Inspektor Gadget directly from its GitHub repository, you can easily install the `kubectl gadget` plug-in and deploy Inspektor Gadget in your cluster using the following commands:

Bash

```
kubectl krew install gadget  
kubectl gadget deploy
```

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Capture a TCP dump from a Linux node in an AKS cluster

Article • 09/27/2024

Networking issues may occur when you're using a Microsoft Azure Kubernetes Service (AKS) cluster. To help investigate these issues, this article explains how to capture a TCP dump from a Linux node in an AKS cluster, and then download the capture to your local machine.

Prerequisites

- The Kubernetes [kubectl](#) tool. To install kubectl by using [Azure CLI](#), run the `az aks install-cli` command.
- An AKS cluster. If you don't have an AKS cluster, [create one using Azure CLI](#) or [through the Azure portal](#).
- The [tcpdump](#) command-line tool installed on the Linux node.

ⓘ Note

You can automate TCP capture through a [Helm chart](#), which can run in the background as a [DaemonSet](#). For more information, see this [custom GitHub tool for capturing TCP dumps](#), or use the steps in the following sections.

Step 1: Find the nodes to troubleshoot

How do you determine which node to pull the TCP dump from? You first get the list of nodes in the AKS cluster using the Kubernetes command-line client, [kubectl](#). Follow the instructions to connect to the cluster and run the `kubectl get nodes --output wide` command [using the Azure portal](#) or [Azure CLI](#). A node list that's similar to the following output appears:

```
Console

$ kubectl get nodes --output wide
NAME                  STATUS   ROLES      AGE     VERSION
INTERNAL-IP    EXTERNAL-IP  OS-IMAGE   KERNEL-VERSION
CONTAINER-RUNTIME
aks-agentpool-34796016-vmss000000  Ready    agent     45h    v1.20.9
10.240.1.81    <none>     Ubuntu    18.04.6 LTS
azure          containerd://1.4.9+azure
```

```
aks-agentpool-34796016-vmss000002 Ready agent 45h v1.20.9  
10.240.2.47 <none> Ubuntu 18.04.6 LTS 5.4.0-1062-  
azure containerd://1.4.9+azure
```

Step 2: Connect to a Linux node

The next step is to establish a connection to the AKS cluster node that you want to capture the network trace from. For more information, see [Create an interactive shell connection to a Linux node](#).

Step 3: Make sure tcpdump is installed

After you've established a connection to the AKS Linux node, verify the `tcpdump` tool has been previously installed on a node by running `tcpdump --version`. If `tcpdump` hasn't been installed, the following error text appears:

```
Console  
  
# tcpdump --version  
bash: tcpdump: command not found
```

Then install `tcpdump` on your pod by running the Advanced Package Tool's package handling utility, [apt-get](#):

```
Bash  
  
apt-get update && apt-get install tcpdump
```

If `tcpdump` is installed, something similar to the following text appears:

```
Console  
  
# tcpdump --version  
tcpdump version 4.9.3  
libpcap version 1.8.1  
OpenSSL 1.1.1 11 Sep 2018
```

Step 4: Create a packet capture

To capture the dump, run the [tcpdump command](#) as follows:

```
Console
```

```
# tcpdump --snapshot-length=0 -vvv -w /capture.cap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144
bytes
Got 6
```

ⓘ Note

Running `tcpdump` without using filtering parameters can significantly increase the size of the Packet Capture (PCAP) file, especially for long runs. Therefore, we recommend that you add filters, such as source, destination, and port. For example:

- `tcpdump dst 192.168.1.100`
- `tcpdump dst host.mydomain.com`
- `tcpdump port http or port ftp or port smtp or port imap or port pop3 or port telnet`

While the trace is running, replicate your issue many times. This action make sure that the issue is captured within the TCP dump. Note the time stamp while you replicate the issue. To stop the packet capture when you're done, press `Ctrl+C`:

Console

```
# tcpdump -s 0 -vvv -w /capture.cap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144
bytes
^C526 packets captured
526 packets received by filter
0 packets dropped by kernel
```

Step 5: Transfer the capture locally

After you complete the packet capture, identify the helper pod so you can copy the dump locally. Open a second console, and then get a list of pods by running `kubectl get pods`, as shown below.

Console

```
$ kubectl get pods
NAME                               READY   STATUS
RESTARTS   AGE
azure-vote-back-6c4dd64bdf-m4nk7   1/1    Running   0
3m29s
azure-vote-front-85b4df594d-jhpzw  1/1    Running   0
```

```
3m29s
node-debugger-aks-nodepool1-38878740-vmss00000-jfsq2    1/1      Running   0
60s
```

The helper pod has a prefix of `node-debugger-aks`, as shown in the third row. Replace the pod name, and then run the following `kubectl` command. These commands retrieve the packet capture for your Linux node.

Bash

```
kubectl cp node-debugger-aks-nodepool1-38878740-vmss00000-jfsq2:/capture.cap capture.cap
```

ⓘ Note

If the `chroot /host` command was used when entering the debug pod, add `/host` before `/capture.cap` for the source file.

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

Yes

No

Capture a TCP dump from a Windows node in an AKS cluster

Article • 10/14/2024

Networking issues may occur when you're using a Microsoft Azure Kubernetes Service (AKS) cluster. To help investigate these issues, this article explains how to capture a TCP dump from a Windows node in an AKS cluster, and then download the capture to your local machine.

Prerequisites

- Azure CLI, version 2.0.59 or later. You can open [Azure Cloud Shell](#) in the web browser to enter Azure CLI commands. Or [install or upgrade Azure CLI](#) on your local machine. To find the version that's installed on your machine, run `az --version`.
- An AKS cluster. If you don't have an AKS cluster, [create one using Azure CLI](#) or [through the Azure portal](#).

Step 1: Find the nodes to troubleshoot

How do you determine which node to pull the TCP dump from? You first get the list of nodes in the AKS cluster using the Kubernetes command-line client, [kubectl](#). Follow the instructions to connect to the cluster and run the `kubectl get nodes --output wide` command [using the Azure portal](#) or [Azure CLI](#). A node list that's similar to the following output appears:

```
Output

$ kubectl get nodes --output wide
NAME           STATUS   ROLES      AGE     VERSION
INTERNAL-IP    EXTERNAL-IP  OS-IMAGE   CONTAINER-RUNTIME
akswin00000    Ready     agent     3m8s    v1.20.9
10.240.0.4    <none>    Windows Server 2019 Datacenter
10.0.17763.2237  docker://20.10.6
akswin00001    Ready     agent     3m50s    v1.20.9
10.240.0.115  <none>    Windows Server 2019 Datacenter
10.0.17763.2237  docker://20.10.6
akswin00002    Ready     agent     3m32s    v1.20.9
10.240.0.226  <none>    Windows Server 2019 Datacenter
10.0.17763.2237  docker://20.10.6
```

Step 2: Connect to a Windows node

The next step is to establish a connection to the AKS cluster node. You authenticate either using a Secure Shell (SSH) key, or using the Windows admin password in a Remote Desktop Protocol (RDP) connection. Both methods require creating an intermediate connection, because you can't currently connect directly to the AKS Windows node. Whether you connect to a node through SSH or RDP, you need to specify the user name for the AKS nodes. By default, this user name is *azureuser*. Besides using an SSH or RDP connection, you can connect to a Windows node from the HostProcess container.

HostProcess

1. Create *hostprocess.yaml* with the following content. Replace `AKSWINDOWSNODENAME` with the AKS Windows node name.

YAML

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    pod: hpc
    name: hpc
spec:
  securityContext:
    windowsOptions:
      hostProcess: true
      runAsUserName: "NT AUTHORITY\\SYSTEM"
  hostNetwork: true
  containers:
    - name: hpc
      image: mcr.microsoft.com/windows/servercore:ltsc2022 # Use
servercore:1809 for WS2019
      command:
        - powershell.exe
        - -Command
        - "Start-Sleep 2147483"
      imagePullPolicy: IfNotPresent
  nodeSelector:
    kubernetes.io/os: windows
    kubernetes.io/hostname: AKSWINDOWSNODENAME
  tolerations:
    - effect: NoSchedule
      key: node.kubernetes.io/unschedulable
      operator: Exists
    - effect: NoSchedule
      key: node.kubernetes.io/network-unavailable
```

```
operator: Exists
- effect: NoExecute
  key: node.kubernetes.io/unreachable
operator: Exists
```

2. Run the `kubectl apply -f hostprocess.yaml` command to deploy the Windows HostProcess container in the specified Windows node.
3. Run the `kubectl exec -it [HPC-POD-NAME] -- powershell` command.
4. Run any PowerShell commands inside the HostProcess container to access the Windows node.

⚠ Note

To access the files in the Windows node, switch the root folder to `C:\` inside the HostProcess container.

Step 3: Create a packet capture

When you're connected to the Windows node through SSH or RDP, or from the HostProcess container, a form of the Windows command prompt appears:

```
Windows Command Prompt
```

```
azureuser@akswin000000 C:\Users\azureuser>
```

Now open a command prompt and enter the [Network Shell](#) (netsh) command below for capturing traces ([netsh trace start](#)). This command starts the packet capture process.

```
Windows Command Prompt
```

```
netsh trace start capture=yes tracefile=C:\temp\AKS_node_name.etl
```

Output appears that's similar to the following text:

```
Output
```

```
Trace configuration:
```

```
-----  
Status:          Running  
Trace File:     AKS_node_name.etl
```

Append:	Off
Circular:	On
Max Size:	250 MB
Report:	Off

While the trace is running, replicate your issue many times. This action ensures the issue has been captured within the TCP dump. Note the time stamp while you replicate the issue. To stop the packet capture when you're done, enter `netsh trace stop`:

Output

```
azureuser@akswin00000 C:\Users\azureuser>netsh trace stop
Merging traces ... done
Generating data collection ... done
The trace file and additional troubleshooting information have been compiled
as "C:\temp\AKS_node_name.cab".
File location = C:\temp\AKS_node_name.etl
Tracing session was successfully stopped.
```

Step 4: Transfer the capture locally

HostProcess

After you complete the packet capture, identify the HostProcess pod so that you can copy the dump locally.

1. On your local machine, open a second console, and then get a list of pods by running the `kubectl get pods` command:

Console

```
kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
azure-vote-back-6c4dd64bdf-m4nk7   1/1     Running   2          3d21h
azure-vote-front-85b4df594d-jhpzw  1/1     Running   2          3d21h
hpc                                 1/1     Running   0          3m58s
```

The HostProcess pod's default name is *hpc*, as shown in the third line.

2. Copy the TCP dump files locally by running the following commands. Replace the pod name with `hpc`.

Console

```
kubectl cp -n default hpc:/temp/AKS_node_name.etl  
./AKS_node_name.etl  
tar: Removing leading '/' from member names  
kubectl cp -n default hpc:/temp/AKS_node_name.etl  
./AKS_node_name.cab  
tar: Removing leading '/' from member names
```

The *.etl* and *.cab* files will now be present in your local directory.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Capture TCP packets from a pod on an AKS cluster

Article • 04/10/2024

This article discusses how to take a TCP traffic capture at a pod of an Azure Kubernetes Service (AKS) cluster, and download the capture to your local computer.

Prerequisite

You must run the Azure CLI version 2.0.59 or a later version.

Run `az --version` to verify the version. To install the latest version of the Azure CLI, see [Install Azure CLI](#).

Identify the pod and install TCPdump

1. Identify the name of the pod that you want to capture the TCP packets from. This should be the pod that has the connectivity issues. To do this, run `kubectl get pods -A` to see the list of pods on your AKS cluster. The following is an example of the output:

Output

NAME	READY	STATUS	RESTARTS	AGE
azure-vote-back-2549686872-4d2r5	1/1	Running	0	31m
azure-vote-front-848767080-tf34m	1/1	Running	0	31m

If you know the namespace that the pod runs in, you also can run `kubectl get pods -n <namespace>` to get a list of pods that are running in that namespace.

2. Connect to the pod that you identified in the previous step. The following commands use "azure-vote-front-848767080-tf34m" as the pod name. Replace them with the correct pod name. If the pod is not in the default namespace, you must add the `--namespace` parameter to the `kubectl exec` command.

Azure CLI

```
kubectl exec azure-vote-front-848767080-tf34m -it -- /bin/bash
```

3. After you connect to the pod, run `tcpdump --version` to determine whether the TCPdump is installed. If you receive a "command not found" message, run the following command to install the TCPdump in the pod:

```
Azure CLI
```

```
apt-get update && apt-get install tcpdump
```

If your pod uses Alpine Linux, run the following command to install TCPdump:

```
Azure CLI
```

```
apk add tcpdump
```

Capture TCP packets and save them to a local directory

1. Run `tcpdump -s 0 -vvv -w /capture.cap` to start capturing TCP packets on your pod.
2. After the packet capture is finished, exit your pod shell session.
3. Run the following command to save the packets to the current directory:

```
Azure CLI
```

```
kubectl cp azure-vote-front-848767080-tf34m:/capture.cap capture.cap
```

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Capture a Windows container dump file from a Windows node in an AKS cluster

Article • 04/26/2024

If a Windows container fails on a Microsoft Azure Kubernetes Service (AKS) cluster, you might have to examine the Windows container dump file to investigate the root cause. This article provides steps to capture a Windows container dump file from a Windows node in an AKS cluster. It also includes instructions to download the dump file to your local computer for further analysis.

Prerequisites

- An AKS cluster. If you don't have an AKS cluster, [create one by using Azure CLI or through the Azure portal](#).
- Windows agent pools that are created after `3/13/2024` or a node image that was upgraded to AKS Windows image version `20240316` or a later version. Alternatively, verify whether the WindowsCSEScriptsPackage version is v0.0.39 or newer, which can be located in `C:\AzureData\CustomDataSetupScript.log` on the Windows nodes.

Step 1: Add annotations metadata to your deployment

Mount a host folder in the container, and add the annotations metadata in order to request that the Windows container store the dump file in a designated folder:

YAML

```
metadata:  
  ...  
annotations:  
  "io.microsoft.container.processdumplocation": "C:\\CrashDumps\\  
{container_id}"  
  "io.microsoft.wcow.processdumpype": "mini"  
  "io.microsoft.wcow.processdumpcount": "10"  
spec:  
  ...  
containers:  
  - name: containername  
    image: ...  
    ...  
volumeMounts:
```

```
- mountPath: C:\CrashDumps
  name: local-dumps
volumes:
- name: local-dumps
  hostPath:
    path: C:\k\containerdumps
    type: DirectoryOrCreate
```

Step 2: Reproduce the issue

Redeploy your deployment, and wait for the Windows container to fail. You can use `kubectl describe pod -n [POD-NAMESPACE] [POD-NAME]` to learn which AKS Windows node is hosting the pod.

Step 3: Connect to the Windows node

Establish a connection to the AKS cluster node. You authenticate either by using a Secure Shell (SSH) key or the Windows admin password in a Remote Desktop Protocol (RDP) connection. Both methods require that you create an intermediate connection. This is because you can't currently connect directly to the AKS Windows node. Whether you connect to a node through SSH or RDP, you have to specify the user name for the AKS nodes. By default, this user name is `azureuser`.

SSH

If you have an SSH key, [create an SSH connection to the Windows node](#). The SSH key doesn't persist on your AKS nodes. The SSH key reverts to what was initially installed on the cluster during any of the following actions:

- Restart
- Version upgrade
- Node image upgrade

Step 4: Transfer the dump file locally

SSH

After the container fails, identify the helper pod so that you can copy the dump file locally. Open a second console, and then get a list of pods by running the `kubectl`

`get pods` command, as follows:

Output

```
kubectl get pods
NAME                               READY   STATUS
RESTARTS   AGE
azure-vote-back-6c4dd64bdf-m4nk7   1/1    Running
2          3d21h
azure-vote-front-85b4df594d-jhpzw 1/1    Running
2          3d21h
node-debugger-aks-nodepool1-38878740-vmss000000-6ztp6 1/1    Running
0          3m58s
```

The helper pod has a prefix of `node-debugger-aks`, as shown in the third row.

Replace the pod name, and then run the following Secure Copy (scp) commands to retrieve the dump files (.dmp) that are saved when the container fails:

Windows Command Prompt

```
scp -o 'ProxyCommand ssh -p 2022 -W %h:%p azureuser@127.0.0.1'
azureuser@10.240.0.97:/C:/k/containerdumps/{container_id}/{application}.
dmp .
```

You can list the `C:\k\containerdumps` folder to find the full path of the dump files after the connection is made to the Windows node.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Errors when mounting an Azure file share

Article • 10/10/2024

This article provides possible causes and solutions for errors that cause the mounting of an Azure file share to fail.

Symptoms

You deploy a Kubernetes resource such as a Deployment or a StatefulSet, in an Azure Kubernetes Service (AKS) environment. The deployment will create a pod that mounts a PersistentVolumeClaim (PVC) referencing an Azure file share.

However, the pod stays in the **ContainerCreating** status. When you run the `kubectl describe pods` command, you might see one of the following errors in the command output, which causes the mounting operation to fail:

- Mount error(2): No such file or directory
- Mount error(13): Permission denied

Refer to the following output as an example:

```
Output

MountVolume.MountDevice failed for volume "\<pv-fileshare-name>"
rpc error: code = Internal desc =
volume(\<storage-account's-resource-group>\#\<storage-account-name>\#\<pv/fileshare-name>\#) > mount "//\<storage-account-name>.file.core.windows.net/\<pv-fileshare-name>" on
"/var/lib/kubelet/plugins/kubernetes.io/csi/pv/\<pv-fileshare-name>/globalmount" failed with
mount failed: exit status 32
Mounting command: mount
Mounting arguments: -t cifs -o
dir_mode=0777,file_mode=0777,uid=0,gid=0,mfsymlinks,cache=strict,actimeo=30,
\<masked> //\<storage-account-name>.file.core.windows.net/\<pv-name>
/var/lib/kubelet/plugins/kubernetes.io/csi/pv/\<pv-name>/globalmount
Output: mount error(\<error-id>): \<error-description>
Refer to the mount.cifs(8) manual page (e.g. man mount.cifs) and kernel log
messages (dmesg)
```

ⓘ Note

- If the storage account is publicly accessible, the hostname displayed in the output will be `<storage-account-name>.file.core.windows.net`.
- If the storage account is configured privately with a private link, endpoint, or DNS zone, the hostname will be `<storage-account-name>.privatelink.file.core.windows.net`.

Before troubleshooting

According to the message in the output, as shown in the following example, identify the storage account and file share—the values will be used in later troubleshooting steps.

```
mount "//<storage-account-name>.file.core.windows.net/<pv-fileshare-name>"
```

See the following sections for possible causes and solutions.

Mount error(2): No such file or directory

This error indicates that there's no connectivity between the AKS cluster and the storage account.

Initial troubleshooting

Azure File relies on SMB protocol (port 445). Make sure that port 445 and/or the IP address of the storage account aren't blocked.

To check the IP address of the storage account, run a Domain Name System (DNS) command like `nslookup`, `dig`, or `host`. For example:

```
Console  
nslookup <storage-account-name>.file.core.windows.net
```

To check if there's connectivity between the AKS cluster and the storage account, get inside the `node` or `pod` and run the following `nc` or `telnet` command:

```
Console  
nc -v -w 2 <storage-account-name>.file.core.windows.net 445
```

Console

```
telnet <storage-account-name>.file.core.windows.net 445
```

Possible causes for mount error(2)

- Cause 1: File share doesn't exist
- Cause 2: NSG blocks traffic between AKS and storage account
- Cause 3: Virtual Appliance blocks traffic between AKS and storage account
- Cause 4: FIPS enabled node pool is used

⚠ Note

- Cause 1, 2, and 4 apply to public and private storage account scenarios.
- Cause 3 applies to the public scenario only.

Cause 1: File share doesn't exist

To check if the file share exists, follow these steps:

1. Search **Storage accounts** in the Azure portal and access your storage account.

Name	Type	Kind	Resource group	Location
c8b	Storage account	StorageV2	cloud-shell-storage-westeurope	West Europe
fb548	Storage account	StorageV2	mc_aks-rg_aks_eastus	East US
fc3	Storage account	StorageV2	mc_aks-rg_aks_eastus	East US
sapiente	Storage account	StorageV2	vmpe-rg	East US

2. Select **File shares** under **Data storage** in the storage account and check if the associated PersistentVolumeClaim in the yaml file of the pod, deployment, or statefulset exists in **File shares**.

Name	Modified	Tier	Quota
pvc-2040577-cf19-49ec-b076-9a422f15053a	4/8/2022, 2:14:48 PM	Transaction optimized	1 GB

Solution: Ensure file share exists

To resolve this issue, make sure that the file share that's associated with the PV/PVC exists.

Cause 2: Network Security Group blocks traffic between AKS and storage account

Check the output of the `nc` or `telnet` command mentioned in the [Initial troubleshooting](#) section. If a timeout is displayed, check the [Network Security Group \(NSG\)](#) and make sure that the IP address of the storage account isn't blocked.

To check if the NSG blocks the IP address of the storage account, follow these steps:

1. In the Azure portal, go to [Network Watcher](#) and select **NSG diagnostic**.
2. Fill the fields by using the following values:
 - **Protocol:** Any
 - **Direction:** Outbound
 - **Source type:** IPv4 address/CIDR
 - **IPv4 address/CIDR:** The IP address of an instance that's associated with the AKS node
 - **Destination IP address:** The IP address of the storage account
 - **Destination port:** 445
3. Select the **Check** button and check the **Traffic** status.

The **Traffic** status can be **Allowed** or **Denied**. The **Denied** status means that the NSG is blocking the traffic between the AKS cluster and storage account. If the status is **Denied**, the NSG name will be shown.

Solution: Allow connectivity between AKS and storage account

To resolve this issue, perform changes accordingly at the NSG level to allow the connectivity between the AKS cluster and the storage account on port 445.

Cause 3: Virtual Appliance blocks traffic between AKS and storage account

If you're using a Virtual Appliance (usually a firewall) to control outbound traffic of the AKS cluster (for example, the Virtual Appliance has a route table applied at the AKS cluster's subnet, and the route table has routes that send traffic towards the Virtual

Appliance), the Virtual Appliance may block traffic between the AKS cluster and the storage account.

To isolate the issue, add a route in the route table for the IP address of the storage account to send the traffic towards the Internet.

To confirm which route table controls the traffic of the AKS cluster, follow these steps:

1. Go to the AKS cluster in the Azure portal and select **Properties > Infrastructure resource group**.
2. Access the virtual machine scale set (VMSS) or a VM in an availability set if you're using such VM set type.
3. Select **Virtual network/subnet > Subnets** and identify the subnet of the AKS cluster. On the right side, you'll see the route table.

To add the route in the route table, follow the steps in [Create a route](#) and fill in the following fields:

- **Address prefix:** <storage-account's-public-IP>/32
- **Next hop type:** Internet

This route will send all traffic between the AKS cluster and storage account through the public Internet.

After you add the route, test the connectivity by using the `nc` or `telnet` command and perform the mounting operation again.

Solution: Ensure Virtual Appliance allows traffic between AKS and storage account

If the mounting operation succeeds, we recommend that you consult your networking team to make sure that the Virtual Appliance can allow traffic between the AKS cluster and storage account on port 445.

Cause 4: FIPS enabled node pool is used

If you use a [Federal Information Processing Standard \(FIPS\) enabled node pool](#), the mounting operation will fail because the FIPS disables some authentication modules, which prevents the mounting of a CIFS share. This behavior is expected and not specific to AKS.

To resolve the issue, use one of the following solutions:

Solution 1: Schedule pods on nodes in a non-FIPS node pool

FIPS is disabled by default on AKS node pools and can be enabled only during the node pool creation by using the `--enable-fips-image` parameter.

To resolve the error, you can schedule the pods on nodes in a non-FIPS node pool.

Solution 2: Create a pod that can be scheduled on a FIPS-enabled node

To create a pod that can be scheduled on a FIPS-enabled node, follow these steps:

1. Use the Azure File CSI driver to create a custom StorageClass that uses NFS protocol.

Refer to the following YAML file as an example:

```
yml

kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: azurefile-sc-fips
provisioner: file.csi.azure.com
reclaimPolicy: Delete
volumeBindingMode: Immediate
allowVolumeExpansion: true
parameters:
  skuName: Premium_LRS
  protocol: nfs
```

The SKU is set to Premium_LRS in the YAML file because Premium SKU is required for NFS. For more information, see [Dynamic Provision](#).

Because of Premium SKU, the minimum size of the file share is 100GB. For more information, see [Create a storage class](#).

2. Create a PVC that references the custom StorageClass `azurefile-sc-fips`.

Refer to the following YAML file as an example:

```
yml

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: azurefile-pvc-fips
spec:
```

```
accessModes:
  - ReadWriteMany
storageClassName: azurefile-sc-fips
resources:
  requests:
    storage: 100Gi
```

3. Create a pod that mounts the PVC *azurefile-pvc-fips*.

Refer to the following YAML file as an example:

```
yml

kind: Pod
apiVersion: v1
metadata:
  name: azurefile-pod-fips
spec:
  containers:
    - name: azurefile-pod-fips
      image: mcr.microsoft.com/oss/nginx/nginx:1.15.5-alpine
      resources:
        requests:
          cpu: 100m
          memory: 128Mi
        limits:
          cpu: 250m
          memory: 256Mi
      volumeMounts:
        - mountPath: "/mnt/azure"
          name: volume
      volumes:
        - name: volume
          persistentVolumeClaim:
            claimName: azurefile-pvc-fips
```

Mount error(13): Permission denied

Here are possible causes for this error:

- Cause 1: Kubernetes secret doesn't reference the correct storage account name or key
- Cause 2: AKS's VNET and subnet aren't allowed for the storage account
- Cause 3: Connectivity is via a private link but nodes and the private endpoint are in different VNETs
- Cause 4: Storage account is set to require encryption that the client doesn't support
- Cause 5: Minimum encryption requirement for a storage account isn't met

- Cause 6: Security profile is used without the NTLM v2 authentication enabled

Note

- Cause 1 applies to public and private scenarios.
- Cause 2 applies to the public scenario only.
- Cause 3 applies to the private scenario only.
- Cause 4 applies to public and private scenarios.
- Cause 5 applies to public and private scenarios.
- Cause 6 applies to public and private scenarios.

Cause 1: Kubernetes secret doesn't reference correct storage account name or key

If the file share is created [dynamically](#), a [Kubernetes secret resource](#) is automatically created with the name "azure-storage-account-<storage-account-name>-secret".

If the file share is created [manually](#), the Kubernetes secret resource should be created manually.

Regardless of the creation method, if the storage account name or the key that's referenced in the Kubernetes secret mismatches the actual value, the mounting operation will fail with the "Permission denied" error.

Possible causes for mismatch

- If the Kubernetes secret is created manually, a typo may occur during the creation.
- If a "Rotate key" operation is performed at the storage account level, the changes won't reflect at the Kubernetes secret level. This will lead to a mismatch between the key value at the storage account level and the value at the Kubernetes secret level.

If a "Rotate key" operation happens, an operation with the name "Regenerate Storage Account Keys" will be displayed in the Activity log of the storage account. Be aware of the [90 days retention period for Activity log](#).

Verify mismatch

To verify the mismatch, follow these steps:

1. Search and access the storage account in the Azure portal. Select **Access keys** > **Show keys** in the storage account. You'll see the storage account name and associated keys.

The screenshot shows the 'Access keys' section of the Azure Storage Account settings for 'fc4...9e264b'. The left sidebar includes links for Overview, Activity log, Tags, Diagnose and solve problems, Access Control (IAM), Data migration, Events, Storage browser (preview), Data storage (Containers, File shares, Queues, Tables), Security + networking (Networking, Azure CDN, Access keys, Shared access signature), and Configuration. The 'Access keys' link is highlighted. The main pane displays two sets of access keys: 'key1' and 'key2'. Each key has a 'Last rotated' timestamp (3/8/2022, 0 days ago), a 'Rotate key' button, and a 'Key' field containing a long, redacted string. Below each key is a 'Connection string' field, also containing a redacted string.

2. Go to the AKS cluster, select **Configuration** > **Secrets**, and then search and access the associated secret.

The screenshot shows the 'Secrets' tab in the AKS Configuration page for the 'aks' cluster. The left sidebar includes links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Security, Kubernetes resources (Namespaces, Workloads, Services and ingresses, Storage, Configuration), and Configuration. The 'Configuration' link is highlighted. The main pane lists a single secret named 'azure-storage-account-fc4...9e264b-secret' under the 'Secrets' tab. It shows the secret's details: Name, Namespace (default), Type (Opaque), Data (2), and Age (14 hours). A search bar at the top allows filtering by secret name, label selector, and namespace.

3. Select **Show** (the eye icon) and compare the values of the storage account name and associated key with the values in Step 1.

The screenshot shows the Azure portal's 'Secrets' section for an AKS cluster. A search bar at the top left contains '9e264b-secret'. On the left, a sidebar lists 'Overview', 'YAML', and 'Events'. On the right, details for the secret are shown: 'Namespace' is 'default', 'Labels' are empty, 'Creation time' is '2022-03-08T21:03:10.000Z', and 'Type' is 'Opaque'. Below this, the 'Data' tab is selected, displaying two key-value pairs. The 'azurstorageaccountkey' value is a long base64 string starting with 'ewogICAgInVzZXJzaG9tQ2VudHJhbF9rZXkiID4gPSJodHRwOi8vZmM...'. The 'azurstorageaccountname' value is another base64 string starting with 'ZmM...'. A magnifying glass icon in the bottom right corner of the data table allows for detailed inspection.

Before you select **Show**, the values of the storage account name and associated key are encoded into base64 strings. After you select **Show**, the values are decoded.

If you don't have access to the AKS cluster in the Azure portal, perform Step 2 at the `kubectl` level:

1. Get the YAML file of the Kubernetes secret, and then run the following command to get the values of the storage account name and the key from the output:

```
Console
kubectl get secret <secret-name> -n <secret-namespace> -o <yaml-file-name>
```

2. Use the `echo` command to decode the values of the storage account name and the key and compare them with the values at the storage account level.

Here's an example to decode the storage account name:

```
Console
echo -n '<storage account name>' | base64 --decode ;echo
azureuser@azure-vm:~$ echo -n 'ZmM...5ZTI2NGI=' | base64 --decode; echo
fc4...9e264b
```

Solution: Adjust the Kubernetes secret and re-create the pods

If the value of the storage account name or key in the Kubernetes secret doesn't match the value in **Access keys** in the storage account, adjust the Kubernetes secret at the Kubernetes secret level by running the following command:

```
Console
```

```
kubectl edit secret <secret-name> -n <secret-namespace>
```

The value of the storage account name or the key added in the Kubernetes secret configuration should be a base64 encoded value. To obtain the encoded value, use the `echo` command.

Here's an example to encode the storage account name:

Console

```
echo -n '<storage account name>' | base64 | tr -d '\n' ; echo
```

For more information, see [Managing Secrets using kubectl](#).

After the Kubernetes secret `azure-storage-account-<storage-account-name>-secret` has the right values, re-create the pods. Otherwise, those pods will continue to use the old values that aren't valid anymore.

Cause 2: AKS's VNET and subnet aren't allowed for storage account

If the storage account's network is limited to selected networks, but the VNET and subnet of the AKS cluster aren't added to selected networks, the mounting operation will fail with the "Permission denied" error.

Solution: Allow AKS's VNET and subnet for storage account

1. Identify the node that hosts the faulty pod by running the following command:

Console

```
kubectl get pod <pod-name> -n <namespace> -o wide
```

Check the node from the command output:

```
azureuser@azuredvm:~$ kubectl get pod nginx-886fd-lkkts -o wide
NAME        READY   STATUS    RESTARTS   AGE     IP          NODE
nginx-886fd-lkkts  0/1    ContainerCreating   0      94s    <none>    aks-nodepool1-2277-vmss000000
```

2. Go to the AKS cluster in the Azure portal, select **Properties > Infrastructure resource group**, access the VMSS associated with the node, and then check **Virtual network/subnet** to identify the VNET and subnet.

Virtual network/subnet : [aks-vnet-117-81/aks-subnet](#)

3. Access the storage account in the Azure portal. Select **Networking**. If **Allow access from** is set to **Selected networks**, check if the VNET and subnet of the AKS cluster are added.

The screenshot shows the 'Networking' tab of a Storage account's settings. Under 'Allow access from', the 'Selected networks' radio button is selected. Below it, there's a section for 'Virtual networks' with a '+ Add existing virtual network' button. On the right side, there's a large circular button with a magnifying glass icon and a '+' sign.

If the VNET and subnet of the AKS cluster aren't added, select **Add existing virtual network**. On the **Add networks** page, type the VNET and subnet of the AKS cluster, and then select **Add > Save**.

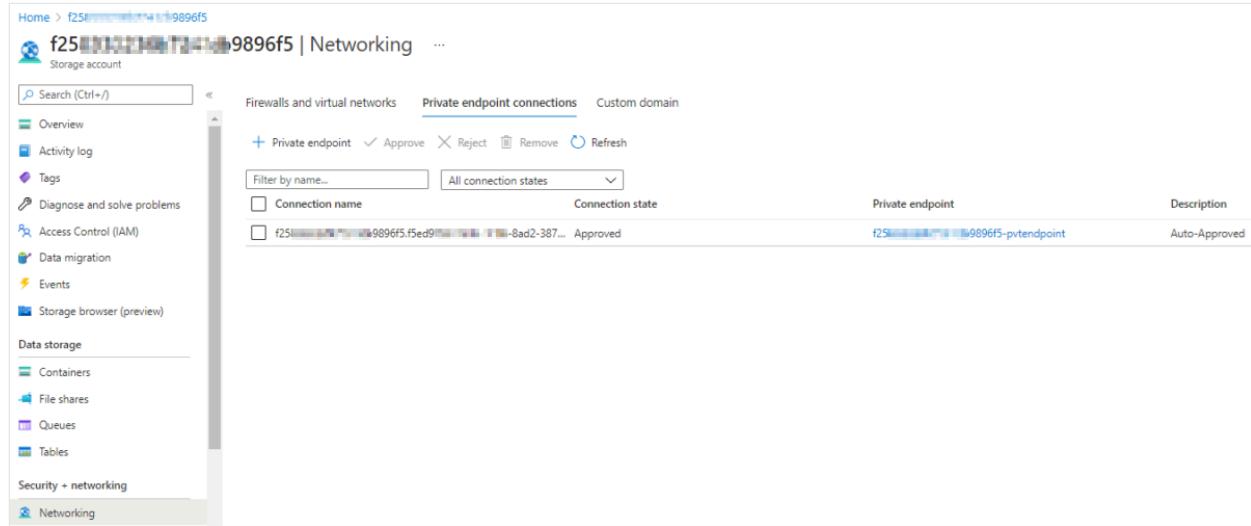
The screenshot shows the 'Add networks' dialog. It has three main sections: 'Subscription' (selected), 'Virtual networks' (selected), and 'Subnets' (selected). Each section has a dropdown menu where 'aks-vnet-117-81' and 'aks-subnet' are chosen, respectively.

It may take a few moments for the changes to take effect. After the VNET and subnet are added, check if the pod status changes from **ContainerCreating** to **Running**.

```
azureuser@azure-vm:~$ kubectl get pod nginx-886fd-lkts -o wide
NAME        READY   STATUS    RESTARTS   AGE     IP           NODE      NOMINATED NODE   READINESS GATES
nginx-886fd-lkts   1/1    Running   0          17m    10.240.0.7   aks-nodepool1-2277-vmss000000   <none>   <none>
```

Cause 3: Connectivity is via private link but nodes and private endpoint are in different VNETs

When the AKS cluster and storage account are connected via a private link, an approved private endpoint connection is used.



The screenshot shows the 'Networking' section of an Azure Storage account. Under 'Private endpoint connections', there is one entry listed:

Connection name	Connection state	Private endpoint	Description
f25...9896f5-5ed9...-8ad2-387...	Approved	f25...9896f5-pvtendpoint	Auto-Approved

In this scenario, if the private endpoint and AKS node are in the same VNET, you'll be able to mount an Azure file share.

If the private endpoint and your AKS cluster are in different VNETs, the mounting operation will fail with the "Permission denied" error.

Solution: Create virtual network link for VNET of AKS cluster at private DNS zone

Get inside the node and check if the fully qualified domain name (FQDN) is resolved via a public or private IP address. To do this, run the following command:

```
Console  
nslookup <storage-account-name>.privatelink.file.core.windows.net
```

If the FQDN is resolved via a public IP address (see the following screenshot), create a virtual network link for the VNET of the AKS cluster at the private DNS zone ("privatelink.file.core.windows.net") level. Note that a virtual network link is already automatically created for the VNET of the storage account's private endpoint.

```
root@aks-nodepool1-16...13-vmss000000:/# nslookup f25...9896f5-5ed9...-8ad2-387...privatelink.file.core.windows.net  
Server: 168.63.129.16  
Address: 168.63.129.16#53  
  
Non-authoritative answer:  
f25...9896f5-5ed9...-8ad2-387...privatelink.file.core.windows.net canonical name = f25...9896f5-store.core.windows.net.  
Name: f25...9896f5-store.core.windows.net  
Address: 20.111.12.100
```

To create the virtual network link, follow these steps:

1. Access the Private DNS zone and select **Virtual network links > Add**.

2. Fill in the fields and select the VNET of the AKS cluster for **Virtual networks**. For how to identify the VNET of the AKS cluster, see the [Solution: Allow AKS's VNET and subnet for storage account](#) section.

3. Select **OK**.

After the virtual network link is added, the FQDN should be resolved via a private IP address, and the mounting operation should succeed. See the following screenshot for an example:

```
root@aks-nodepool1-16...13-vmss000000:/# nslookup f25...9896f5.privatefile.core.windows.net
Server:      168.63.129.16
Address:     168.63.129.16#53

Non-authoritative answer:
Name:   f25...9896f5.privatefile.core.windows.net
Address: 172.17.0.5
```

Cause 4: Storage account is set to require encryption that the client doesn't support

Azure Files Security Settings contain a number of options for controlling the security and encryption settings on storage accounts. Restricting allowed methods and algorithms can prevent clients from connecting.

AKS versions earlier than 1.25 are based on Ubuntu 18.04 LTS, which uses the Linux 5.4 kernel and only supports the AES-128-CCM and AES-128-GCM encryption algorithms. The **Maximum security** profile, or a **Custom** profile that disables AES-128-GCM, will cause share mapping failures.

AKS versions 1.25 and later versions are based on Ubuntu 22.04, which uses the Linux 5.15 kernel and has support for AES-256-GCM.

Solution: Allow AES-128-GCM encryption algorithm to be used

Enable the AES-128-GCM algorithm by using the **Maximum compatibility** profile or a **Custom** profile that enables AES-128-GCM. For more information, see [Azure Files Security Settings](#).

Cause 5: Minimum encryption requirement for a storage account isn't met

Solution: Enable AES-128-GCM encryption algorithm for all storage accounts

To successfully mount or access a file share, the AES-128-GCM encryption algorithm should be enabled for all storage accounts.

If you want to use the AES-256-GCM encryption only, do the following:

Linux

Use the following script to check if the client supports AES-256-GCM and enforce it only if it does:

Bash

```
cifsConfPath="/etc/modprobe.d/cifs.conf"
echo "$(date) before change ${cifsConfPath}:"
cat ${cifsConfPath}
```

```

# Check if 'require_gcm_256' is already present in the configuration file
if ! grep -q "require_gcm_256" "${cifsConfPath}"; then

    # Load the CIFS module
    modprobe cifs

    # Set the parameter at runtime
    echo 1 > /sys/module/cifs/parameters/require_gcm_256

    # Persist the configuration
    echo "options cifs require_gcm_256=1" >> "${cifsConfPath}"

    echo "$(date) after changing ${cifsConfPath}:"
    cat "${cifsConfPath}"
else
    echo "require_gcm_256 is already set in ${cifsConfPath}"
fi

```

You can also use a Kubernetes DaemonSet to enforce AES-256 on every node. See the following example:

[support-cifs-aes-256-gcm.yaml ↗](#)

Windows

Use the [Set-SmbClientConfiguration](#) PowerShell command to specify the encryption ciphers used by the SMB client and the preferred encryption type without user confirmation:

PowerShell

```
Set-SmbClientConfiguration -EncryptionCiphers "AES_256_GCM" -Confirm:$false
```

ⓘ Note

The `EncryptionCiphers` parameter is available beginning with the 2022-06 Cumulative Update for Windows Server version 21H2 for x64-based systems ([KB5014665 ↗](#)) and the Cumulative Update for Windows 11, version 22H2 ([KB5014668 ↗](#)).

Cause 6: Security profile is used without the NTLM v2 authentication enabled

When you use the **Maximum security** profile or a **Custom** security profile without the **NTLM v2** authentication mechanism enabled, the mounting operation will fail with the "Mount error(13): Permission denied" error.

Solution: Enable the NTLM v2 authentication or use the "Maximum compatibility" profile

To mount it properly in AKS, you have to enable the **NTLM v2** authentication mechanism for the **Custom** security profile or use the **Maximum compatibility** security profile.

More information

If you experience some other mount errors, see [Troubleshoot Azure Files problems in Linux](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?



Errors when mounting an Azure Blob storage container

Article • 09/06/2024

This article provides possible causes and solutions for errors that cause the mounting of an Azure Blob storage container to fail.

Symptoms

In an Azure Kubernetes Service (AKS) environment, you deploy a Kubernetes resource such as a Deployment and a StatefulSet. The deployment will create a pod that mounts a PersistentVolumeClaim (PVC) referencing an Azure Blob storage container.

However, the pod stays in the `ContainerCreating` status. When you run the `kubectl describe pods` command, you may see one of the following errors in the command output, which causes the mounting operation to fail.

ⓘ Note

Azure Blob storage container may use the BlobFuse or Network File System (NFS) version 3.0 protocol. BlobFuse and NFS 3.0 scenarios are documented separately in this article.

BlobFuse related errors

- BlobFuse error 1: Exit status 255. Unable to start blobfuse, errno=404. Unable to start blobfuse due to authentication or connectivity issues
- BlobFuse error 2: Exit status 255. Unable to start blobfuse, errno=403. Unable to start blobfuse due to authentication or connectivity issues
- BlobFuse error 3: Context deadline exceeded/An operation with the given Volume ID <...> already exists

NFS 3.0 related errors

- NFS 3.0 error 1: Exit status 32. No such file or directory
- NFS 3.0 error 2: Exit status 32, access denied by server while mounting
- NFS 3.0 error 3: Context deadline exceeded/An operation with the given Volume ID <...> already exists

See the following sections for possible causes and solutions.

ⓘ Note

Although the cause of an issue may be similar in some scenarios, the error may differ because of the different nature of the BlobFuse and NFS 3.0 protocols.

BlobFuse error 1: Exit status 255 Unable to start blobfuse, errno=404, Unable to start blobfuse due to authentication or connectivity issues

Cause: The Blob container doesn't exist

To check if the Blob container exists, follow these steps:

1. Search for Storage accounts in the Azure portal and access your storage account.

The screenshot shows the Azure Storage accounts page. It lists two storage accounts: 'fuse7' and 'nfs8'. Both are of type 'Storage account' and are 'BlockBlobStorage'. They are located in the 'mc_aks-blob-rg_aks-blob_westeurope' resource group in 'West Europe'. There is a search bar at the top and filter buttons for Subscription, Resource group, and Location.

Name	Type	Kind	Resource group	Location
fuse7	Storage account	BlockBlobStorage	mc_aks-blob-rg_aks-blob_westeurope	West Europe
nfs8	Storage account	BlockBlobStorage	mc_aks-blob-rg_aks-blob_westeurope	West Europe

2. Select Containers under Data storage in the storage account and check if the associated PersistentVolume (PV) exists in Containers. To see the Persistent Volume (PV), check the Persistent Volume Claim (PVC) associated with the pod in the YAML file, and then check which PV is associated with that PVC.

The screenshot shows the 'Containers' section of the Azure Storage account 'fuse7'. It displays a single container named 'pvc-c'. The container was last modified on 12/3/2022, 12:34:40 PM, has a Private public access level, and is Available in terms of lease state. The left sidebar shows other options like Overview, Activity log, Tags, and Data storage.

Solution: Ensure the container exists

To resolve this issue, make sure that the Blob container that's associated with the PV/PVC exists.

BlobFuse error 2: Exit status 255 Unable to start blobfuse, errno=403, Unable to start blobfuse due to authentication or connectivity issues

Here are the possible causes for this error:

- Cause 1: Kubernetes secret doesn't reference the correct storage account name or key
- Cause 2: AKS's virtual network (VNET) and subnet aren't allowed for the storage account

Cause 1: Kubernetes secret doesn't reference the correct storage account name or key

If the Blob storage container is created [dynamically](#), a Kubernetes secret resource is automatically created with the name "azure-storage-account-<storage-account-name>-secret".

If the Blob storage container is created [manually](#), the Kubernetes secret resource should be created manually.

Regardless of the creation method, if the storage account name or the key that's referenced in the Kubernetes secret mismatches the actual value, the mounting operation will fail with the "Permission denied" error.

Possible causes for the mismatch

- If the Kubernetes secret is created manually, a typo may occur during the creation.
- If a "Rotate key" operation is performed at the storage account level, the changes won't be reflected at the Kubernetes secret level. This will lead to a mismatch between the key value at the storage account level and the value at the Kubernetes secret level.

If a "Rotate key" operation happens, an operation with the name "Regenerate Storage Account Keys" will be displayed in the Activity log of the storage account. Be aware of the [90 days retention period for Activity log](#).

Verify the mismatch

To verify the mismatch, follow these steps:

1. Search for and access the storage account in the Azure portal. Select **Access keys** > **Show keys** in the storage account. You'll see the storage account name and associated keys.

The screenshot shows the 'Access keys' section of the Azure Storage account settings for 'fuse7...'. It displays two sets of keys: 'key1' and 'key2'. Each key includes a 'Rotate key' button, a note about the last rotation (12/2/2022), and a 'Show' button to view the key value. Below each key is a 'Connection string' input field with a 'Show' button. The left sidebar shows other storage account management options like Overview, Activity log, Tags, and Data migration.

2. Go to the AKS cluster, select **Configuration** > **Secrets**, and then search for and access the associated secret.

The screenshot shows the 'Secrets' tab in the AKS cluster configuration. It lists a single secret named 'azure-storage-account-fuse7...-d-secret' under the 'Name' column. The 'Namespace' is 'default' and the 'Type' is 'Opaque'. The secret was created 2 minutes ago. The left sidebar shows other Kubernetes resources like Namespaces, Workloads, Services and ingresses, and Storage.

3. Select **Show** (the eye icon) and compare the values of the storage account name and associated key with the values in step 1.

The screenshot shows the Azure portal's 'Secrets' blade for a Kubernetes secret named 'fuse7'. The secret is of type 'Opaque' and was created on 2022-12-02T13:39:45.000Z. It contains two data entries: 'azurstorageaccountkey' and 'azurstorageaccountname'. The values for both entries are displayed as long, illegible base64 strings. A magnifying glass icon is overlaid on the secret name 'fuse7'.

Before you select **Show**, the values of the storage account name and associated key are encoded into base64 strings. After you select **Show**, the values are decoded.

If you don't have access to the AKS cluster in the Azure portal, perform step 2 at the `kubectl` level:

1. Get the YAML file of the Kubernetes secret, and then run the following command to get the values of the storage account name and the key from the output:

```
Console
kubectl get secret <secret-name> -n <secret-namespace> -o <yaml-file-name>
```

2. Use the `echo` command to decode the values of the storage account name and the key and compare them with the values at the storage account level.

Here's an example of decoding the storage account name:

```
Console
echo -n '<storage account name>' | base64 --decode ;echo
```

```
azureuser@azure:~$ echo -n 'ZnVzaWQuc2FycG9ybmFtZS5jb20i | base64 --decode ;echo
fuse7'
```

Solution: Adjust the Kubernetes secret and re-create the pods

If the value of the storage account name or key in the Kubernetes secret doesn't match the value in **Access keys** in the storage account, adjust the Kubernetes secret at the Kubernetes secret level by running the following command:

Console

```
kubectl edit secret <secret-name> -n <secret-namespace>
```

The value of the storage account name or the key added in the Kubernetes secret configuration should be a base64 encoded value. To get the encoded value, use the `echo` command.

Here's an example of encoding the storage account name:

Console

```
echo -n '<storage account name>' | base64 | tr -d '\n' ; echo
```

For more information, see [Managing Secrets using kubectl](#).

After the Kubernetes secret `azure-storage-account-<storage-account-name>-secret` has the correct values, re-create the pods. Otherwise, those pods will continue to use the old values that aren't valid anymore.

Cause 2: AKS's VNET and subnet aren't allowed for the storage account

If the storage account's network is limited to selected networks, but the VNET and subnet of the AKS cluster aren't added to the selected networks, the mounting operation will fail.

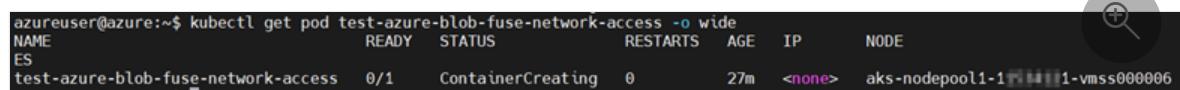
Solution: Allow AKS's VNET and subnet for the storage account

1. Identify the node that hosts the faulty pod by running the following command:

Console

```
kubectl get pod <pod-name> -n <namespace> -o wide
```

Check the node in the command output:



NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
test-azure-blob-fuse-network-access	0/1	ContainerCreating	0	27m	<none>	aks-nodepool1-1-vmss000006

2. Go to the AKS cluster in the Azure portal, select **Properties > Infrastructure resource group**, access the virtual machine scale set (VMSS) associated with the node, and then check the **Virtual network/subnet** to identify the VNET and subnet.

Virtual network/subnet : [aks-vnet-3](#) / [aks-subnet](#)

3. Access the storage account in the Azure portal, and then select **Networking**. If **Public network access** is set to **Enabled from selected virtual networks** or **Disabled**, and the connectivity isn't through a private endpoint, check if the VNET and subnet of the AKS cluster are allowed under **Firewalls and virtual networks**.

If the VNET and subnet of the AKS cluster aren't added, select **Add existing virtual network**. On the **Add networks** page, type the VNET and subnet of the AKS cluster, and then select **Add > Save**.

It may take a few moments for the changes to take effect. After the VNET and subnet are added, check if the pod status changes from **ContainerCreating** to **Running**.

```
azureuser@azure:~$ kubectl get pod test-azure-blob-fuse-network-access -o wide
NAME           READY   STATUS    RESTARTS   AGE   IP          NODE
test-azure-blob-fuse-network-access   1/1    Running   0          39m   10.244.6.2   aks-nodepool1-1-vmss000006
```

BlobFuse error 3: Context deadline exceeded/An operation with the given Volume ID <...> already exists

Here are possible causes for this error:

- Cause 1: Network Security Group blocks traffic between AKS and the storage account
- Cause 2: Virtual Appliance blocks traffic between AKS and the storage account

Initial troubleshooting for BlobFuse error 3

Azure Blob BlobFuse relies on port 443. Make sure that port 443 and/or the IP address of the storage account aren't blocked.

To check the IP address of the storage account, run a Domain Name System (DNS) command like `nslookup`, `dig`, or `host`. For example:

```
Console  
nslookup <storage-account-name>.blob.core.windows.net
```

To check if there's connectivity between the AKS cluster and the storage account, get inside the `node` or `pod` and run the following `nc` or `telnet` command:

```
Console  
nc -v -w 2 <storage-account-name>.blob.core.windows.net 443
```

```
Console  
telnet <storage-account-name>.blob.core.windows.net 443
```

Cause 1: Network Security Group blocks traffic between AKS and the storage account

Check the output of the `nc` or `telnet` command mentioned in the [Initial troubleshooting for BlobFuse error 3](#) section. If a time-out is displayed, check the [Network Security Group \(NSG\)](#) and make sure that the IP address of the storage account isn't blocked.

To check if the NSG blocks the IP address of the storage account, follow these steps:

1. In the Azure portal, go to [Network Watcher](#) and select **NSG diagnostic**.
2. Fill in the fields by using the following values:
 - **Protocol:** TCP
 - **Direction:** Outbound
 - **Source type:** IPv4 address/CIDR
 - **IPv4 address/CIDR:** The IP address of an instance that's associated with the AKS node
 - **Destination IP address:** The IP address of the storage account
 - **Destination port:** 443 (if using BlobFuse) and 111/2048 (if using NFS)
3. Select the **Check** button and check the **Traffic** status.

The **Traffic** status can be **Allowed** or **Denied**. The **Denied** status means that the NSG is blocking the traffic between the AKS cluster and the storage account. If the status is **Denied**, the NSG name will be shown.

Solution: Allow connectivity between AKS and the storage account

To resolve this issue, make changes accordingly at the NSG level to allow the connectivity between the AKS cluster and the storage account on port 443.

Cause 2: Virtual Appliance blocks traffic between AKS and the storage account

If you're using a Virtual Appliance (usually a firewall) to control outbound traffic of the AKS cluster (for example, the Virtual Appliance has a route table applied at the AKS cluster's subnet, and the route table has routes that send traffic towards the Virtual Appliance), the Virtual Appliance may block traffic between the AKS cluster and the storage account.

To isolate the issue, add a route in the route table for the IP address of the storage account to send the traffic towards the Internet.

To confirm which route table controls the traffic of the AKS cluster, follow these steps:

1. Go to the AKS cluster in the Azure portal and select **Properties > Infrastructure resource group**.

2. Access the VMSS or a virtual machine (VM) in an availability set if you're using such a VM set type.
3. Select **Virtual network/subnet > Subnets** and identify the subnet of the AKS cluster. On the right side, you'll see the route table.

To add a route in the route table, follow the steps in [Create a route](#) and fill in the following fields:

- **Address prefix:** <storage-account's-public-IP>/32
- **Next hop type:** Internet

This route will send all traffic between the AKS cluster and the storage account through the public internet.

After you add the route, test the connectivity by using the `nc` or `telnet` command and perform the mounting operation again.

Solution: Ensure Virtual Appliance allows traffic between AKS and the storage account

If the mounting operation succeeds, we recommend that you consult your networking team to make sure that the Virtual Appliance can allow traffic between the AKS cluster and the storage account on port 443.

NFS 3.0 error 1: Exit status 32. No such file or directory

Cause: Blob container doesn't exist

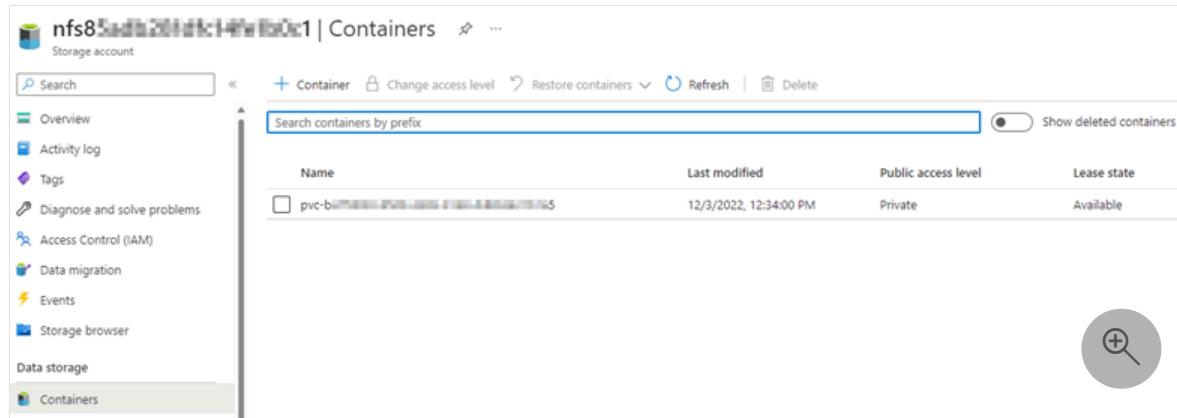
To check if the Blob container exists, follow these steps:

1. Search for **Storage accounts** in the Azure portal and access your storage account.

Name	Type	Kind	Resource group	Location
fuse7	Storage account	BlockBlobStorage	mc_aks-blob-rg_aks-blob_westeurope	West Europe
nfs8	Storage account	BlockBlobStorage	mc_aks-blob-rg_aks-blob_westeurope	West Europe

2. Select **Containers** under **Data storage** in the storage account and check if the associated PersistentVolume (PV) exists in **Containers**. To see the Persistent Volume

(PV), check the Persistent Volume Claim (PVC) associated with the pod in the YAML file, and then check which PV is associated with that PVC.



Solution: Ensure the Blob container exists

To resolve this issue, make sure that the Blob container that's associated with the PV/PVC exists.

NFS 3.0 error 2: Exit status 32, access denied by server while mounting

Cause: AKS's VNET and subnet aren't allowed for the storage account

If the storage account's network is limited to selected networks, but the VNET and subnet of the AKS cluster aren't added to the selected networks, the mounting operation will fail.

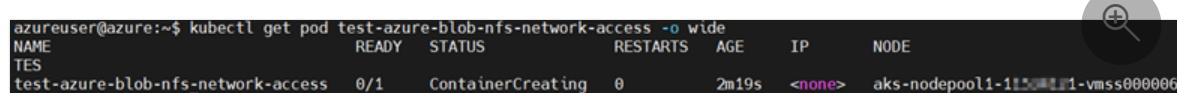
Solution: Allow AKS's VNET and subnet for the storage account

1. Identify the node that hosts the faulty pod by running the following command:

```
Console

kubectl get pod <pod-name> -n <namespace> -o wide
```

Check the node in the command output:



NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
test-azure-blob-nfs-network-access	0/1	ContainerCreating	0	2m19s	<none>	aks-nodepool1-115-111-1-vmss000006

2. Go to the AKS cluster in the Azure portal, select **Properties > Infrastructure resource group**, access the VMSS associated with the node, and then check the **Virtual network/subnet** to identify the VNET and subnet.

Virtual network/subnet : [aks-vnet-3](#) [!0/aks-subnet](#)

3. Access the storage account in the Azure portal, and then select **Networking**. If **Public network access** is set to **Enabled from selected virtual networks** or **Disabled**, and the connectivity isn't through a private endpoint, check if the VNET and subnet of the AKS cluster are allowed under **Firewalls and virtual networks**.

Virtual Network	Subnet	Address range	Endpoint Status	Resource Group	Subscription
No network selected.					

If the VNET and subnet of the AKS cluster aren't added, select **Add existing virtual network**. On the **Add networks** page, type the VNET and subnet of the AKS cluster, and then select **Add > Save**.

Add networks

Subscription *

A [dropdown] Subscription

Virtual networks *

aks-vnet-3

Subnets *

aks-subnet

It may take a few moments for the changes to take effect. After the VNET and subnet are added, check if the pod status changes from **ContainerCreating** to **Running**.

```
azureuser@azure:~$ kubectl get pod test-azure-blob-nfs-network-access -o wide
NAME                  READY   STATUS    RESTARTS   AGE     IP           NODE
test-azure-blob-nfs-network-access   1/1    Running   0          5m15s   10.244.6.3   aks-nodepool1-1-vmss000006
```

NFS 3.0 error 3: context deadline exceeded / An operation with the given Volume ID <...> already exists

Here are possible causes for this error:

- Cause 1: Network Security Group blocks traffic between AKS and the storage account
- Cause 2: Virtual Appliance blocks traffic between AKS and the storage account

Initial troubleshooting for NFS 3.0 error 3

Azure Blob NFS 3.0 relies on ports 111 and 2049. Make sure that ports 111 and 2049 and/or the IP address of the storage account aren't blocked.

To check the IP address of the storage account, run a Domain Name System (DNS) command like `nslookup`, `dig`, or `host`. For example:

```
Console
```

```
nslookup <storage-account-name>.blob.core.windows.net
```

To check if there's connectivity between the AKS cluster and the storage account, get inside the `node` or `pod` and run the following `nc` or `telnet` command:

```
Console
```

```
nc -v -w 2 <storage-account-name>.blob.core.windows.net 111
```

```
Console
```

```
nc -v -w 2 <storage-account-name>.blob.core.windows.net 2049
```

```
Console
```

```
telnet <storage-account-name>.blob.core.windows.net 111
```

```
Console
```

```
telnet <storage-account-name>.blob.core.windows.net 2049
```

Cause 1: Network Security Group blocks traffic between AKS and the storage account

Check the output of the `nc` or `telnet` command mentioned in the [Initial troubleshooting for NFS 3.0 error 3](#) section. If a time-out is displayed, check the [Network Security Group \(NSG\)](#) and make sure that the IP address of the storage account isn't blocked.

To check if the NSG blocks the IP address of the storage account, follow these steps:

1. In the Azure portal, go to [Network Watcher](#) and select **NSG diagnostic**.
2. Fill in the fields by using the following values:
 - **Protocol:** TCP
 - **Direction:** Outbound
 - **Source type:** IPv4 address/CIDR
 - **IPv4 address/CIDR:** The IP address of an instance that's associated with the AKS node
 - **Destination IP address:** The IP address of the storage account
 - **Destination port:** 111 then 2048
3. Select the **Check** button and check the **Traffic** status.

The **Traffic** status can be **Allowed** or **Denied**. The **Denied** status means that the NSG is blocking the traffic between the AKS cluster and the storage account. If the status is **Denied**, the NSG name will be shown.

Solution: Allow connectivity between AKS and the storage account

To resolve this issue, make changes accordingly at the NSG level to allow the connectivity between the AKS cluster and the storage account on port 111/2048.

Cause 2: Virtual Appliance blocks traffic between AKS and the storage account

If you're using a Virtual Appliance (usually a firewall) to control outbound traffic of the AKS cluster (for example, the Virtual Appliance has a route table applied at the AKS cluster's subnet, and the route table has routes that send traffic towards the Virtual Appliance), the Virtual Appliance may block traffic between the AKS cluster and the storage account.

To isolate the issue, add a route in the route table for the IP address of the storage account to send the traffic towards the Internet.

To confirm which route table controls the traffic of the AKS cluster, follow these steps:

1. Go to the AKS cluster in the Azure portal and select **Properties > Infrastructure resource group**.
2. Access the virtual machine scale set (VMSS) or a VM in an availability set if you're using such a VM set type.
3. Select **Virtual network/subnet > Subnets** and identify the subnet of the AKS cluster. On the right side, you'll see the route table.

To add a route in the route table, follow the steps in [Create a route](#) and fill in the following fields:

- **Address prefix:** <storage-account's-public-IP>/32
- **Next hop type:** Internet

This route will send all traffic between the AKS cluster and the storage account through the public internet.

After you add the route, test the connectivity by using the `nc` or `telnet` command and perform the mounting operation again.

Solution: Ensure Virtual Appliance allows traffic between AKS and the storage account

If the mounting operation succeeds, we recommend that you consult your networking team to make sure that the Virtual Appliance can allow traffic between the AKS cluster and the storage account on ports 111 and 2049.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?



Yes



No

Errors when mounting Azure disk volumes

Article • 04/07/2025

This article provides solutions for errors that cause the mounting of Azure disk volumes to fail.

Symptoms

You're trying to deploy a Kubernetes resource, such as a Deployment or a StatefulSet, in an Azure Kubernetes Service (AKS) environment. The deployment creates a pod that should mount a PersistentVolumeClaim (PVC) that references an Azure disk.

However, the pod stays in the **ContainerCreating** status. When you run the `kubectl describe pods` command, you may see one of the following errors that cause the mounting operation to fail:

- Disk cannot be attached to the VM because it is not in the same zone as the VM
- Client '<client-ID>' with object id '<object-ID>' doesn't have authorization to perform action over scope '<disk name>' or scope is invalid
- Volume is already used by pod
- StorageAccountType UltraSSD_LRS can be used only when additionalCapabilities.ultraSSDEnabled is set
- ApplyFSGroup failed for vol
- Node(s) exceed max volume count

See the following sections for error details, possible causes, and solutions.

Disk cannot be attached to the VM because it is not in the same zone as the VM

Here are details of this error:

Output

```
AttachVolume.Attach failed for volume "<disk/PV name>":  
rpc error:  
code = Unknown  
desc = Attach volume "/subscriptions/<subscription-ID>/resourceGroups/<disk-  
resource-group>/providers/Microsoft.Compute/disks/<disk/PV name>" to instance "<AKS node name>" failed with Retriable: false, RetryAfter: 0s, HttpStatusCode:  
400,  
RawError:  
{  
    "error":  
    {
```

```
"code": "BadRequest",
"message": "Disk /subscriptions/<subscription-ID>/resourceGroups/<disk-resource-group>/providers/Microsoft.Compute/disks/<disk/PV name > cannot be attached to the VM because it is not in the same zone as the VM. VM zone: 'X'. Disk zone: 'Y'. "
}
```

Cause: Disk and node hosting pod are in different zones

In AKS, the default and other built-in storage classes for Azure disks use [locally redundant storage \(LRS\)](#). These disks are deployed in [availability zones](#). If you use the node pool in AKS together with availability zones, and the pod is scheduled on a node that's in another availability zone that's different from the disk, you might experience this error.

To resolve this error, use one of the following solutions.

Solution 1: Ensure disk and node hosting the pod are in the same zone

To make sure that the disk and node that host the pod are in the same availability zone, use [node affinity](#).

Refer to the following script as an example:

```
yml

affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
        - matchExpressions:
          - key: topology.disk.csi.azure.com/zone
            operator: In
            values:
              - <region>-Y
```

<region> is the region of the AKS cluster. Y represents the availability zone of the disk (for example, westeurope-3).

Solution 2: Use zone-redundant storage (ZRS) disks

ZRS disk volumes can be scheduled on all zone and non-zone agent nodes. For more information, see [Azure disk availability zone support](#).

To use a ZRS disk, create a storage class by using `Premium_ZRS` or `StandardSSD_ZRS`, and then deploy the PersistentVolumeClaim (PVC) that references the storage.

For more information about parameters, see [Driver Parameters](#)

Solution 3: Use Azure Files

[Azure Files](#) is mounted by using NFS or SMB throughout network. It's not associated with availability zones.

For more information, see the following articles:

- [Dynamically create Azure Files share](#)
- [Manually create Azure Files share](#)

Client '<client-ID>' with object id '<object-ID>' doesn't have authorization to perform action over scope '<disk name>' or scope is invalid

Here are details of this error:

```
Output

AttachVolume.Attach failed for volume "pv-azuredisk":
rpc error: code = NotFound
desc = Volume not found, failed with error: Retriable: false, RetryAfter: 0s,
HTTPStatusCode: 403,
RawError:
{
  "error": {
    "code": "AuthorizationFailed",
    "message": "The client '<client-ID>' with object id '<object-ID>' does not have
    authorization to perform action 'Microsoft.Compute/disks/read' over scope
    '/subscriptions/<subscription-ID>/resourceGroups/<disk-resource-
    group>/providers/Microsoft.Compute/disks/<disk name>' or the scope is invalid. If
    access was recently granted, please refresh your credentials."
  }
}
```

Cause: AKS identity doesn't have required authorization over disk

AKS cluster's identity doesn't have the required authorization over the Azure disk. This issue occurs if the disk is created in a resource group other than the infrastructure resource group of

the AKS cluster.

Solution: Create role assignment that includes required authorization

Create a role assignment that includes the authorization required per the error. We recommend that you use a [Contributor](#) role. If you want to use another built-in role, see [Azure built-in roles](#).

To assign a Contributor role, use one of the following methods:

- [Use the az role assignment create command](#)

Here's an example:

Azure CLI

```
az role assignment create --assignee <AKS-identity-ID> --role "Contributor" -  
-scope /subscriptions/<subscription-ID>/resourceGroups/<disk-resource-  
group>/providers/Microsoft.Compute/disks/<disk name>
```

- Use the Azure portal

Refer to detailed steps that are introduced in [Assign Azure roles using the Azure portal](#) to assign a Contributor role. If you assign the Contributor role to a managed identity, use the [control plane identity](#) to manage Azure disks.

Volume is already used by pod

Here are details of this error:

Multi-Attach error for volume "<PV/disk-name>" Volume is already used by pod(s) <pod-name>

Cause: Disk is mounted to multiple pods hosted on different nodes

An Azure disk can be mounted only as [ReadWriteOnce](#). This makes it available to one node in AKS. That means that it can be attached to only one node and mounted to only a pod that's hosted by that node. If you mount the same disk to a pod on another node, you experience this error because the disk is already attached to a node.

Solution: Make sure disk isn't mounted by multiple pods hosted on different nodes

To resolve this error, refer to [Multi-Attach error](#).

To share a PersistentVolume across multiple nodes, use [Azure Files](#).

StorageAccountType UltraSSD_LRS can be used only when additionalCapabilities.ultraSSDEnabled is set

Here are details of this error:

```
Output

AttachVolume.Attach failed for volume "<Disk/PV name>" :
rpc error:
code = Unknown
desc = Attach volume "/subscriptions/<subscription-ID>/resourceGroups/<disk-
resource-group>/providers/Microsoft.Compute/disks/<disk/PV name>" to instance "<AKS node name>" failed with Retriable: false, RetryAfter: 0s, HttpStatusCode:
400, RawError:
{
  "error": {
    "code": "InvalidParameter",
    "message": "StorageAccountType UltraSSD_LRS can be used only when
additionalCapabilities.ultraSSDEnabled is set.",
    "target": "managedDisk.storageAccountType"
  }
}
```

Cause: Ultra disk is attached to node pool with ultra disks disabled

This error indicates that an [ultra disk](#) is trying to be attached to a node pool by having ultra disks disabled. By default, an ultra disk is disabled on AKS node pools.

Solution: Create a node pool that can use ultra disks

To use ultra disks on AKS, create a node pool that has ultra disks support by using the `--enable-ultra-ssd` flag. For more information, see [Use Azure ultra disks on Azure Kubernetes Service](#).

ApplyFSGroup failed for vol

Here are details of this error:

```
MountVolume.SetUp failed for volume '<PV/disk-name>': applyFSGroup failed for vol  
/subscriptions/<subscriptionID>/resourceGroups/<infra/disk-resource-  
group>/providers/Microsoft.Compute/disks/<PV/disk-name>: [...]: input/output error
```

Cause: Changing ownership and permissions for large volume takes much time

If there are many files already present in the volume, and if a `securityContext` that uses `fsGroup` exists, this error might occur. If there are lots of files and directories in one volume, changing the group ID would consume excessive time. Additionally, the Kubernetes official documentation [Configure volume permission and ownership change policy for Pods](#) mentions this situation:

"By default, Kubernetes recursively changes ownership and permissions for the contents of each volume to match the `fsGroup` specified in a Pod's `securityContext` when that volume is mounted. For large volumes, checking and changing ownership and permissions can take much time, slowing Pod startup. You can use the `fsGroupChangePolicy` field inside a `securityContext` to control the way that Kubernetes checks and manages ownership and permissions for a volume."

Solution: Set `fsGroupChangePolicy` field to `OnRootMismatch`

To resolve this error, we recommend that you set `fsGroupChangePolicy: "OnRootMismatch"` in the `securityContext` of a Deployment, a StatefulSet, or a pod.

`OnRootMismatch`: Change permissions and ownership only if permission and ownership of the root directory doesn't match the expected permissions of the volume. This setting could help shorten the time that it takes to change ownership and permission of a volume.

For more information, see [Configure volume permission and ownership change policy for Pods](#).

Node(s) exceed max volume count

Here are details of this error:

Output

Events:

Type	Reason	Age	From	Message
Warning	FailedScheduling	25s	default-scheduler	0/8 nodes are available: 8 node(s) exceed max volume count. preemption: 0/8 nodes are available: 8 No preemption victims found for incoming pod..

Cause: Maximum disk limit is reached

The node has reached its maximum disk capacity. In AKS, the number of disks per node depends on the VM size that's configured for the node pool.

Solution

To resolve the issue, use one of the following methods:

- Add a new node pool with a VM size that supports more disk limit.
- Scale the node pool.
- Delete existing disks from the node.

Additionally, make sure that the number of disks per node does not exceed the [Kubernetes default limits](#).

More information

For more Azure Disk known issues, see [Azure disk plugin known issues](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Can't set the uid and gid mounting options on an Azure disk

Article • 10/22/2024

This article discusses how to fix issues that occur on a Microsoft Azure Kubernetes Service (AKS) cluster when you try to set the mounting options for user identifier (`uid`) and group identifier (`gid`) on an Azure disk.

Symptoms

If you try to set `uid` and `gid` to 1000 in the `mountOptions` setting of the Azure disk's storage class configuration, you receive an error that resembles the following text:

```
Output

Warning FailedMount          63s           kubelet, aks-
nodepool1-29460110-0 MountVolume.MountDevice failed for volume "pvc-
d783d0e4-85a1-11e9-8a90-369885447933" : azureDisk -
mountDevice:FormatAndMount failed with mount failed: exit status 32
Mounting command: systemd-run
Mounting arguments: --description=Kubernetes transient mount for
/var/lib/kubelet/plugins/kubernetes.io/azure-disk/mounts/m436970985 --scope
-- mount -t xfs -o dir_mode=0777,file_mode=0777,uid=1000,gid=1000,defaults
/dev/disk/azure/scsi1/lun2 /var/lib/kubelet/plugins/kubernetes.io/azure-
disk/mounts/m436970985
Output: Running scope as unit run-rb21966413ab449b3a242ae9b0fbc9398.scope.
mount: wrong fs type, bad option, bad superblock on /dev/sde,
      missing codepage or helper program, or other error
```

Cause

Azure disk uses the ext4/xfs filesystem by default, and mounting options such as `uid=x,gid=x` can't be set at mount time.

Workaround 1: Use a pod security context to specify the user and group

Configure the security context for a pod [↗](#). In the pod configuration file (YAML format), within the `spec/securityContext` field, set the `runAsUser` field to the previously

specified `uid` value, and set the `fsGroup` field to the previously specified `gid` value. For example, the beginning of the configuration file might resemble the following code:

YAML

```
apiVersion: v1
kind: Pod
metadata:
  name: security-context-demo
spec:
  securityContext:
    runAsUser: 0
    fsGroup: 0
```

ⓘ Note

By default, the `gid` and `uid` options are mounted as root or 0. If those options are set as non-root (such as 1000), Kubernetes will use the `change owner` command (`chown`) to change all directories and files within that disk. This operation can be time-consuming, and it might make the disk-mounting operation very slow.

Workaround 2: Use an init container and the `change owner` command to specify the user and group

Set up an [init container](#), and then run a `chown` command that specifies the user and group values in the container. The following code shows how to set up an init container to set the user and group:

YAML

```
initContainers:
- name: volume-mount
  image: mcr.microsoft.com/aks/fundamental/base-ubuntu:v0.0.11
  command: ["sh", "-c", "chown -R 100:100 /data"]
  volumeMounts:
  - name: <your-data-volume>
    mountPath: /data
```

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about

the performance or reliability of these products.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

"Could not change permissions" error while using Azure Files

Article • 02/28/2025

This article discusses how to troubleshoot a "could not change permissions" error when you use Microsoft Azure Files with your Azure Kubernetes Service (AKS) cluster.

Symptoms

When you run PostgreSQL on the Azure Files plug-in, you receive an error that resembles the following output:

```
initdb: could not change permissions of directory "/var/lib/postgresql/data":  
Operation not permitted  
  
fixing permissions on existing directory /var/lib/postgresql/data  
  
Message: 'OSError while changing ownership of the log file. 'Arguments:  
PermissionError: [Errno 1] Operation not permitted
```

Cause

The Azure Files plug-in uses the Common Internet File System (CIFS) protocol, which is a dialect of the Server Message Block (SMB) protocol. When this protocol is in use, the file and directory permissions can't be changed after the files and directories are mounted.

Workaround

Use the Azure Disk plug-in instead, and [use the subPath property ↗](#).

Note

For the ext3 or ext4 disk type, there's a *lost+found* directory after the disk is formatted.

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about

the performance or reliability of these products.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

Failed to create file share on storage account

Article • 03/14/2025

This article discusses how to troubleshoot why you can't create a file share on a storage account that's used for dynamic provisioning on Azure Kubernetes Service (AKS).

Symptoms

When you create a file share on a storage account that's used for dynamic provisioning, the PersistentVolumeClaim (PVC) is stuck in the Pending status. In this case, if you run the `kubectl describe pvc` command, you receive the following error:

`persistentvolume-controller` (combined from similar events):

Failed to provision volume with StorageClass "azurefile":

failed to create share kubernetes-dynamic-pvc-xxx in account xxx:

failed to create file share, err:

storage: service returned error: StatusCode=403, ErrorCode=AuthorizationFailure,

ErrorMessage=This request is not authorized to perform this operation.

Cause

The Kubernetes `persistentvolume-controller` isn't on the network that was chosen when the **Allow access from** network setting was enabled for **Selected networks** on the storage account. Especially, when you specify `useDataPlaneAPI: "true"` on the storage class, the `persistentvolume-controller` uses the data plane API for file share creation/deletion/resizing. However, this will fail when a firewall or virtual network is set on the storage account.

Workaround

Create a file share and set up your AKS cluster to use [static provisioning with Azure Files](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

File share mounting failures for Azure Files

07/10/2025

This article discusses how to troubleshoot file share mounting failures for Azure Files so that you can set up storage successfully on your Microsoft Azure Kubernetes Service (AKS) clusters.

Common cause

Your storage account key has changed. This problem can cause a range of errors. The following pod error is typical:

```
Error: failed to generate container  
"56907e9807c6f4203c3aace8c5a6e3a75832cf07d3080a3869d355114657b54f" spec: failed to  
generate spec: failed to stat "/var/lib/kubelet/pods/xxxxxxxxx-9fe6-46d7-b12e-  
339951b8d2f5/volumes/kubernetes.io~csi/pvc-xxxxxxxxx-3b70-498c-b357-  
3488e1c1f429/mount": stat /var/lib/kubelet/pods/xxxxxxxxx-9fe6-46d7-b12e-  
339951b8d2f5/volumes/kubernetes.io~csi/pvc-xxxxxxxxx-3b70-498c-b357-  
3488e1c1f429/mount: host is down
```

Solution

Manually update the `azurestorageaccountkey` field in an Azure file secret to add your base64-encoded storage account key. To make this update, follow these steps:

1. Encode your storage account key in base64 by running the following command:

```
Console  
  
echo <storage-account-key> | base64
```

2. Update your Azure secret file [↗](#) by running the `kubectl edit secret` command to open the secret file in your default text editor:

```
Console  
  
kubectl edit secret azure-storage-account-<storage-account-name>-secret
```

3. Change the `azurestorageaccountkey` field to use the new base64-encoded storage account key, and then save the file.

4. Redeploy your pods.

Note

Simply deleting the pod and allowing it to be re-created might not resolve the issue. Make sure that you redeploy the pod.

After a few minutes, the agent node will retry the Azure File mount operation by using the updated storage key.

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Use mountOptions settings in Azure Files

Article • 04/27/2025

This article discusses recommended mount options when you configure the storage class object on Azure Files. These mounting options help you to provision storage on your Kubernetes cluster.

Recommended settings

The following `mountOptions` settings are recommended for Server Message Block (SMB) and Network File System (NFS) shares:

- **SMB shares**

YAML

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: azurefile-csi
provisioner: file.csi.azure.com
allowVolumeExpansion: true
parameters:
  skuName: Premium_LRS # available values: Premium_LRS, Premium_ZRS,
Standard_LRS, Standard_GRS, Standard_ZRS, Standard_RAGRS, Standard_RAGZRS
reclaimPolicy: Delete
volumeBindingMode: Immediate
mountOptions:
  - dir_mode=0777 # modify this permission if you want to enhance the
    security
  - file_mode=0777 # modify this permission if you want to enhance the
    security
  - mfsymlinks # support symbolic links
  - cache=strict # https://linux.die.net/man/8/mount.cifs
  - nosharesock # reduces probability of reconnect race
  - actimeo=30 # reduces latency for metadata-heavy workload
  - nobrl # disable sending byte range lock requests to the server and for
    applications which have challenges with posix locks
```

- **NFS shares**

YAML

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: azurefile-csi-nfs
provisioner: file.csi.azure.com
```

```
parameters:
  protocol: nfs
  skuName: Premium_LRS      # available values: Premium_LRS, Premium_ZRS
  reclaimPolicy: Delete
  volumeBindingMode: Immediate
  allowVolumeExpansion: true
  mountOptions:
    - nconnect=4  # improves performance by enabling multiple connections to share
    - noresvport  # improves availability
    - actimeo=30  # reduces latency for metadata-heavy workloads
```

① Note

The location for configuring mount options (`mountOptions`) depends on whether you provision dynamic or static persistent volumes. If you [dynamically provision a volume](#) with a storage class, specify the mount options on the storage class object (`kind: StorageClass`). If you [statically provision a volume](#), specify the mount options on the `PersistentVolume` object (`kind: PersistentVolume`). If you [mount the file share as an inline volume](#), specify the mount options on the `Pod` object (`kind: Pod`).

More information

For Azure Files best practices, see [Provision Azure Files storage](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Slow attach and detach operations for an Azure disk

Article • 03/24/2025

This article discusses how to troubleshoot slow attach and detach operations when you use an Azure disk for storage on your Microsoft Azure Kubernetes Service (AKS) clusters.

Symptoms

Attaching and detaching Azure disks takes more time than expected if there are many operations, as outlined in the following table.

[] Expand table

Target of attach and detach operations	Number of attach and detach operations
Single-node virtual machine	More than 10
Single virtual machine scale set pool	More than 3

Cause

This limitation is a known issue that affects the in-tree Azure disk driver (`kubernetes.io/azure-disk`) that performs the operations sequentially.

To verify that you're using the in-tree Azure Disk driver, check the storage class that's defined on the persistent volume claim. If the `provisioner` is `kubernetes.io/azure-disk`, then you're using the in-tree driver. This means that you're exposed to the referred limitation in which operations are done sequentially.

Usage of the in-tree driver was deprecated in AKS 1.21. We now use the Container Storage Interface (CSI) driver as the standard driver.

Workaround

We recommend that you [migrate your in-tree volumes to CSI Drivers by using this documentation](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

Troubleshoot common Bring Your Own Key (BYOK) issues

Article • 04/10/2024

This article discusses troubleshooting methods for the most common Bring Your Own Key (BYOK) issues on Microsoft Azure Kubernetes Service (AKS).

Prerequisites

- [Azure CLI](#)

Troubleshooting checklist

Step 1: Register a preview feature to enable BYOK on an ephemeral OS disk

To register the preview feature that enables BYOK on an ephemeral operating system (OS) disk (`Microsoft.ContainerService/EnableBYOKOnEphemeralOSDiskPreview`), run the [az feature register](#) and [az feature show](#) commands. For instructions, see [Register customer-managed key \(preview\) feature](#).

Step 2: Create a cluster that uses BYOK on an ephemeral OS disk

To create an AKS cluster that uses BYOK on an ephemeral OS disk, run the [az aks create](#) command. For instructions, see [Create a new AKS cluster and encrypt the OS disk](#).

Note

If you don't specify the `--node-osdisk-type` parameter in the [az aks create](#) command, the default OS disk type for the node is `Ephemeral` only if the current virtual machine (VM) size supports ephemeral OS disks. For more information, see [Ephemeral OS disks for Azure VMs](#).

Cause 1: Can't enable purge protection in the key vault

If you receive a `CreateVMSSAgentPoolFailed` error code and a `KeyVaultNotPurgeProtectionEnabled` error subcode, the creation of an agent pool for a virtual machine scale set failed because you didn't enable purge protection when you created the key vault.

Solution 1: Enable purge protection when you create the key vault

Run the `az keyvault create` command again, and specify a value of `true` for the `--enable-purge-protection` parameter. For instructions, see [Create an Azure Key Vault instance](#).

Cause 2: Key vault and disk are in different regions when agent pool creation fails for a virtual machine scale set

If you receive a `CreateVMSSAgentPoolFailed` error code and a `KeyVaultAndDiskInDifferentRegions` error subcode, the key vault and the disk encryption set are located in different regions. The key vault and disk encryption set must be in the same region.

Solution 2: Use the same region as for the disk when you create the key vault

Run the `az keyvault create` command again, and make sure that the value that you specify in the `--location` parameter is the same region that you used for the disk encryption set. For instructions, see [Create an Azure Key Vault instance](#).

Cause 3: Disk encryption set isn't granted

If you delete the disk encryption set permission for the key vault, the AKS cluster node won't run. In this situation, you receive the following error message in the disk encryption set page of the Azure portal:

To associate a disk, image, or snapshot with this disk encryption set, you must grant permissions to the key vault.

Solution 3: Update the security policy settings

Update the security policy settings again by running the [az keyvault set-policy](#) command. For instructions, see [Grant the DiskEncryptionSet access to key vault](#).

Reference

- Bring your own keys (BYOK) with Azure disks in Azure Kubernetes Service (AKS)

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?



Troubleshoot pods and namespaces stuck in the Terminating state

Article • 05/10/2024

This article discusses troubleshooting strategies for a scenario in Microsoft Azure Kubernetes Service (AKS) in which pods and namespaces remain stuck in the `Terminating` state.

Prerequisites

- The Kubernetes [kubectl](#) tool.

Note: To install kubectl by using [Azure CLI](#), run the `az aks install-cli` command.

Troubleshooting checklist

Step 1: Determine which pod to delete

Verify the name of the pod that you have to remove and the namespace that the pod belongs to. To determine which pods are running on your AKS cluster and the namespaces that the pods are operating in, run the following [kubectl get](#) command:

```
Bash
```

```
kubectl get pod --all-namespaces
```

Step 2: Delete the pod

Using the information from Step 1, run the following [kubectl delete](#) command to delete the pod:

```
Bash
```

```
kubectl delete <pod-name> --namespace <namespace-name>
```

 **Note**

You can omit the `--namespace <namespace-name>` parameter if the specified pod belongs to the "default" namespace.

If you receive the following error message, make sure that the pod name and the namespace name are correct:

```
Error from server (NotFound): pods "<POD NAME>" not found
```

If the pod and namespace names are correct, but the pod wasn't deleted, you can forcibly delete the pod. To do this, run the following `kubectl delete` command:

Bash

```
kubectl delete pod <pod-name> --namespace <namespace-name> --grace-period=0  
--force --wait=false
```

Step 3: Determine which namespace to delete

Verify the name of the namespace that you have to remove. To determine which namespaces are running on your AKS cluster, run the following `kubectl get` command:

Bash

```
kubectl get namespace
```

Step 4: Find resources within the namespace

If a namespace is stuck in the `Terminating` state, find all the resources that are defined within the namespace. To do this, run the following `kubectl get` command:

Bash

```
kubectl get all --namespace <namespace-name>
```

Step 5: Delete resources within the namespace

After you discover which resources are defined within the namespace, delete those resources. For each resource to delete, run the following `kubectl delete` command:

Bash

```
kubectl delete <resource> <resource name> --namespace <namespace-name> --grace-period=0 --force --wait=false
```

For example, if you want to delete the `nginxtest` pod within the `nginx` namespace, run the following command:

Bash

```
kubectl delete pod nginxtest --namespace nginx --grace-period=0 --force --wait=false
```

Step 6: Delete the namespace

After you delete all the resources within the namespace, delete the namespace itself. To do this, run the following `kubectl delete` command:

Bash

```
kubectl delete namespace <namespace-name> --grace-period=0 --force --wait=false
```

⚠ Warning

The `kubectl delete` command might not be successful initially if you [use finalizers to prevent accidental deletion](#). Finalizers are keys on resources that signal pre-delete operations. Finalizers control the garbage collection on resources, and they're designed to alert controllers about what cleanup operations to do before they remove a resource.

However, finalizers don't necessarily identify code that should be executed. In fact, finalizers resemble annotations in the following manner:

- They are basically lists of keys.
- They can be manipulated.

If you try to delete a resource that has a finalizer on it, the resource remains in finalization until the controller removes the finalizer keys or the finalizers are removed by using `kubectl`. After the finalizer list is emptied, Kubernetes can reclaim the resource and put it into a queue to be deleted from the registry.

If no resources remain in the namespace, but the namespace is still stuck in the `Terminating` state, run the following [kubectl patch](#) command to empty the finalizer field:

Bash

```
kubectl patch namespace <namespace-name> --patch '{"metadata": {"finalizers": null}}'
```

This action enables you to delete the namespace successfully when you run the `kubectl delete` command again.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Troubleshoot seccomp profile configuration in Azure Kubernetes Service

06/06/2025

Secure computing (seccomp) [↗](#) is a Linux kernel feature that restricts the system calls (syscalls) containers can make, enhancing the security of containerized workloads. In Azure Kubernetes Service (AKS), the [containerd](#) [↗](#) runtime used by AKS nodes natively supports seccomp. Enabling a seccomp profile might cause AKS workloads to fail because workload critical syscalls are blocked. This article introduces what seccomp profiles are, how they work, and how to troubleshoot them using the open source project [Inspektor Gadget](#) [↗](#).

Background

Syscalls are the interface that allows user space programs to request kernel services. Seccomp profiles specify the syscalls that are allowed or denied for a specific container. AKS supports two values:

- `RuntimeDefault`: Use the default seccomp profile given by the runtime.
- `Unconfined`: All syscalls are allowed.

To enable seccomp on your AKS node pools, see [Secure container access to resources using built-in Linux security features](#). You can also configure a custom profile to meet your workload's specific needs, see [Configure a custom seccomp profile](#) for details.

When using seccomp profiles, it's important to test and validate the effect on your workloads. Some workloads might require a lower number of syscall restrictions than others. This means that if workloads require syscalls that aren't included in the configured profile, they might fail during runtime.

This article shows how to use the open source project [Inspektor Gadget](#) [↗](#) to diagnose issues and gain visibility into blocked syscalls.

Symptoms

After you configure AKS workloads to use a seccomp profile, the workloads exit unexpectedly with one of the following errors:

- permission denied
- function not implemented

Prerequisites

- A tool to connect to the Kubernetes cluster, such as the `kubectl` tool. To install `kubectl` using the [Azure CLI](#), run the `az aks install-cli` command.
- The [krew](#) package manager for installing [Inspektor Gadget](#)'s plugin. You can follow the [krew quickstart guide](#) to install it.
- The seccomp profile that you're trying to troubleshoot.
- The open source project [Inspektor Gadget](#).

Troubleshooting checklist

Step 1: Modify your seccomp profile

Create a custom seccomp profile matching the one you're troubleshooting and replace its default action such as `SCMP_ACT_ERRNO` with `SCMP_ACT_LOG` to log blocked syscalls instead of failing them.

Your custom seccomp profile might look like this:

JSON

```
{  
    "defaultAction": "SCMP_ACT_ALLOW",  
    "syscalls": [  
        {  
            "names": [ "acct",  
                      "add_key",  
                      "bpf",  
                      "clock_adjtime",  
                      "clock_settime",  
                      "clone",  
                      "create_module",  
                      "delete_module",  
                      "finit_module",  
                      "get_kernel_syms",  
                      "get_mempolicy",  
                      "init_module",  
                      "ioperm",  
                      "iopl",  
                      "kcmp",  
                      "kexec_file_load",  
                      "kexec_load",  
                      "keyctl",  
                      "lookup_dcookie",  
                      "mbind",  
                      "mount",  
                      "move_pages",  
                      "perf_event_open",  
                      "pivot_root",  
                      "readahead_sets",  
                      "remap_file_pages",  
                      "set_memory_policy",  
                      "setns",  
                      "sigreturn",  
                      "splice",  
                      "sync_file_range",  
                      "syncfs",  
                      "tunables",  
                      "vmsplice" ]  
        }  
    ]  
}
```

```
        "nfsservctl",
        "open_by_handle_at",
        "perf_event_open",
        "personality",
        "pivot_root",
        "process_vm_readv",
        "process_vm_writev",
        "ptrace",
        "query_module",
        "quotactl",
        "reboot",
        "request_key",
        "set_mempolicy",
        "setsns",
        "settimeofday",
        "stime",
        "swapon",
        "swapoff",
        "sysfs",
        "_sysctl",
        "umount",
        "umount2",
        "unshare",
        "uselib",
        "userfaultfd",
        "ustat",
        "vm86",
        "vm86old"],
    "action": "SCMP_ACT_LOG"
}
]
}
```

The article [Configure a custom seccomp profile](#) shows how you can apply your custom seccomp profile to your AKS cluster. Alternatively, you can follow these steps:

1. Get the names of the nodes in your AKS cluster by running the following command:

Console

```
kubectl get nodes
```

2. Use the `kubectl debug` command to start a debug pod on the node and make sure the `seccomp` folder exists and the `tar` tool is installed (for copying the profile into the node in the next step):

Console

```
kubectl debug node/<node-name> -it --
image=mcr.microsoft.com/azurelinux/base/core:3.0
```

```
root [ / ]# mkdir -p /host/var/lib/kubelet/seccomp
root [ / ]# tdnf install -y tar
```

3. Copy the pod name printed when running the `kubectl debug` command. It looks like `node-debugger-<node-name>-<random-suffix>`. It can also be retrieved by listing the pods in the `default` namespace.

4. In another terminal, transfer the seccomp profile file to the node directly:

Console

```
kubectl cp <the path of the local seccomp profile>/my-profile.json <pod name>:/host/var/lib/kubelet/seccomp/my-profile.json
```

 Note

Repeat the preceding steps for each node in the cluster to ensure that the seccomp profile is available on all nodes where your workload might run.

Now, you can modify the `seccompProfile` specification of the target pod, which should be confined to the recorded syscalls. For example:

YAML

```
apiVersion: v1
kind: Pod
metadata:
  name: default-pod
  labels:
    app: default-pod
spec:
  securityContext:
    seccompProfile:
      type: Localhost
      localhostProfile: my-profile.json
  containers:
    - name: test-container
      image: docker.io/library/nginx:latest
```

Step 2: Install Inspektor Gadget

[Inspektor Gadget](#) provides insights into syscalls affecting your containers. To use it, run the following commands to install the `gadget` kubectl plugin in your host and deploy [Inspektor Gadget](#) into the cluster:

Console

```
kubectl krew install gadget
```

Console

```
kubectl gadget deploy
```

For more information, see [How to install Inspektor Gadget in an AKS cluster](#).

Step 3: Run the audit_seccomp gadget

With [Inspektor Gadget](#) installed, start the [audit_seccomp gadget](#) using the [kubectl gadget run command](#):

Console

```
kubectl gadget run audit_seccomp
```

Step 4: Analyze blocked syscalls

Run your workload using the `kubectl apply -f` command. Then, the [audit_seccomp gadget](#) logs the syscalls that the seccomp profile should block, along with their associated pods, containers, and processes. You can use this information to identify the root causes of workload failures.

For example, if you run the above-mentioned `default-pod` pod with the `my-profile.json` profile, the output looks like the following one:

Output

K8S.NODE	K8S.NAMESPACE	K8S.PODNAME	K8S.CONTAINERNAME
COMM	PID	TID	CODE SYSCALL
aks-nodepool1-38695788-vmss00002			default default-pod test-container
docker-entrypoi	3996610	3996610	SECCOMP_RET_LOG SYS_CLONE
aks-nodepool1-38695788-vmss00002			default default-pod test-container
docker-entrypoi	3996610	3996610	SECCOMP_RET_LOG SYS_CLONE
aks-nodepool1-38695788-vmss00002			default default-pod test-container
docker-entrypoi	3996610	3996610	SECCOMP_RET_LOG SYS_CLONE
aks-nodepool1-38695788-vmss00002			default default-pod test-container
docker-entrypoi	3996610	3996610	SECCOMP_RET_LOG SYS_CLONE
aks-nodepool1-38695788-vmss00002			default default-pod test-container
docker-entrypoi	3996610	3996610	SECCOMP_RET_LOG SYS_CLONE
aks-nodepool1-38695788-vmss00002			default default-pod test-container
10-listen-on-ip	3996628	3996628	SECCOMP_RET_LOG SYS_CLONE

```

aks-nodepool1-38695788-vmss00002 default      default-pod test-container
10-listen-on-ip 3996628 3996628 SECCOMP_RET_LOG SYS_CLONE
aks-nodepool1-38695788-vmss00002 default      default-pod test-container
10-listen-on-ip 3996632 3996632 SECCOMP_RET_LOG SYS_CLONE
aks-nodepool1-38695788-vmss00002 default      default-pod test-container
10-listen-on-ip 3996632 3996632 SECCOMP_RET_LOG SYS_CLONE
aks-nodepool1-38695788-vmss00002 default      default-pod test-container
10-listen-on-ip 3996632 3996632 SECCOMP_RET_LOG SYS_CLONE
aks-nodepool1-38695788-vmss00002 default      default-pod test-container
10-listen-on-ip 3996632 3996632 SECCOMP_RET_LOG SYS_CLONE
aks-nodepool1-38695788-vmss00002 default      default-pod test-container
10-listen-on-ip 3996632 3996632 SECCOMP_RET_LOG SYS_CLONE
aks-nodepool1-38695788-vmss00002 default      default-pod test-container
20-envsubst-on- 3996639 3996639 SECCOMP_RET_LOG SYS_CLONE
aks-nodepool1-38695788-vmss00002 default      default-pod test-container
20-envsubst-on- 3996639 3996639 SECCOMP_RET_LOG SYS_CLONE
aks-nodepool1-38695788-vmss00002 default      default-pod test-container
20-envsubst-on- 3996641 3996641 SECCOMP_RET_LOG SYS_CLONE
aks-nodepool1-38695788-vmss00002 default      default-pod test-container
30-tune-worker- 3996643 3996643 SECCOMP_RET_LOG SYS_CLONE
aks-nodepool1-38695788-vmss00002 default      default-pod test-container
nginx          3996610 3996610 SECCOMP_RET_LOG SYS_CLONE
aks-nodepool1-38695788-vmss00002 default      default-pod test-container
nginx          3996610 3996610 SECCOMP_RET_LOG SYS_CLONE

```

The output indicates that the `test-container` executes the `SYS_CLONE` syscall that the seccomp profile should block. With this information, you can determine whether to permit the listed syscalls in your container. If so, adjust the seccomp profile by removing them, which prevents the workload from failing.

Here are some commonly blocked syscalls to watch out for. A more comprehensive list is available in [Significant syscalls blocked by default profile](#).

[] [Expand table](#)

Blocked syscall	Consideration
<code>clock_settime</code> or <code>clock_adjtime</code>	If your workload needs accurate time synchronization, ensure this syscall isn't blocked.
<code>add_key</code> or <code>key_ctl</code>	If your workload requires key management, these blocked syscalls prevent containers from using the kernel keyring that is used for retaining security data, authentication keys, encryption keys, and other data within the kernel.
<code>clone</code>	This syscall prevents the cloning of new namespaces, except for <code>CLONE_NEWUSER</code> . Workloads that depend on creating new namespaces might be affected if this syscall is blocked.
<code>io_uring</code>	This syscall is blocked with the move to <code>containerd</code> 2.0. However, it's not blocked in the profile for <code>containerd</code> 1.7.

Next steps

If you encounter issues with your workloads due to blocked syscalls, consider using a custom seccomp profile suited to the specific needs of your application. You can check out the [Inspektor Gadget advise_seccomp gadget](#).

Testing and refining seccomp profiles helps maintain performance and security for AKS workloads. For further assistance, see [Secure computing](#).

Related content

[Secure container access to resources using built-in Linux security features](#)

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the Azure Policy Add-on for Azure Kubernetes Service (AKS)

Article • 08/07/2024

This article contains a list of resources that can help you recover from issues that occur when you use the Microsoft Azure Policy Add-on in conjunction with an Azure Kubernetes Service (AKS) cluster.

Before you begin

Review the [official guide to troubleshooting Kubernetes clusters](#). There's also a [troubleshooting guide on GitHub](#) that's published by a Microsoft engineer for troubleshooting pods, nodes, clusters, and other features.

Related content

The following articles address specific issues that you might encounter when you use the Azure Policy Add-on for AKS:

- [AKS container CPU and memory limits aren't enforced](#)
- [AKS upgrade fails when using an Azure custom policy](#)
- [Azure Policy can't exclude privileged containers from AKS clusters](#)
- [Azure policy failure alert shows an empty list of affected resources](#)
- [Azure policy reports 'ConstraintNotProcessed' for running container](#)
- [Custom Azure Policy for validating controllers doesn't work](#)
- [Pods are created in the user namespaces](#)
- [Pods fail and restart after you enable Defender Profile for AKS](#)
- ['Policy definition not found' during policy assignments in Terraform](#)
- [Retain privileged mode while adding capabilities](#)
- [Security best practice: Don't run as root in containers](#)
- [Third-party scanner detects run-time host incident](#)

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

AKS upgrade fails when using an Azure custom policy

Article • 04/24/2024

This article discusses how to resolve a scenario in which a Kubernetes upgrade fails when you use the Microsoft Azure Policy Add-on for Azure Kubernetes Service (AKS).

Symptoms

You see a message in the Azure portal that indicates that your AKS cluster is compliant with your Azure custom policy. However, when you try to upgrade AKS, the custom policy blocks the upgrade attempt.

Cause

During an AKS upgrade, the policies that are defined for the upgrade might differ from the policies that are defined for a running cluster. This scenario can create policy-related issues that appear only during the upgrade process. The scenario might occur if you start the upgrade process by using Azure CLI.

Solution

Try using the [Azure portal](#) instead of Azure CLI to start the upgrade.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Security best practice: Don't run as root in containers

Article • 06/23/2024

To improve security, we recommend that you don't run as a root user inside containers that are hosted on Azure Kubernetes Service. To run the container as a nonroot user, specify the following `securityContext` settings in the YAML file when you deploy a pod or other Azure Kubernetes resources.

SecurityContext

- Resource: Pod / Deployment / DaemonSet / StatefulSet / ReplicaSet / ReplicationController / Job / CronJob
- Arguments:
 - **runAsNonRoot (Optional)**: If it's true, the container operates without root privileges. Default is false.
 - **runAsUser (Optional)**: If user number is anything other than 0 (root), the container runs by using that user ID (not the root user).

By default, the `securityContext` field is `empty ({})`. To implement these fields in the YAML file, see [Configure a security context for a pod or container](#). After you add these configurations, redeploy the pods to enforce the updates. If the `securityContext` field is omitted, the pod runs as root.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes	 No
---	--

Azure policy failure alert shows an empty list of affected resources

Article • 04/12/2024

This article provides a solution to an issue where a policy failure alert shows an empty list of affected resources when you use the Azure Policy Add-on for Azure Kubernetes Service (AKS).

Symptoms

When an Azure policy deployed in an AKS cluster fails, the policy failure alert incorrectly displays an empty list of affected resources in the Azure portal.

Cause

This issue might be caused by a delay or Graph API issue. This issue can occur even if the policy violation has been resolved.

Resolution

If you suspect the alert is erroneous and the policy violation has been mitigated, wait for some time to allow for potential updates and synchronizations. In most cases, the issue can be resolved automatically when the system updates the alert status.

However, if the alert persists for more than a week or you believe the alert is inaccurate, we recommend opening a support case with Azure Support. Provide details about the specific policy, the related alert, error messages, or any relevant information. The Azure Support team will assist in investigating and resolving the issue.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Azure Policy can't exclude privileged containers from AKS clusters

Article • 04/12/2024

This article provides a solution to an issue where Azure Policy can't exclude privileged containers from Azure Kubernetes Service (AKS) clusters.

Symptoms

When you use Azure Policy in an AKS cluster, excluding privileged containers fails.

Cause

Azure's built-in policies don't have container exclusion parameters for AKS clusters.

Resolution

To resolve this issue, exclude containers from Azure Policy enforcement within your cluster. Here are the steps:

1. Create an exemption in the default Azure Security Center policy. This exemption allows you to customize the policy to meet specific requirements.

When creating an exemption, you can apply it to the whole assignment or exempt specific assignments based on your requirements. For more information, see [Exempt a resource](#).

2. Create a new policy assignment that excludes the desired pods or containers. This overrides the default policy behavior and excludes the specified pods or containers from policy enforcement.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Azure policy reports 'ConstraintNotProcessed' for running container

Article • 04/10/2024

This article provides a solution to the error that you encounter when Azure Policy reports a running container as **ConstraintNotProcessed**.

Symptoms

A container such as Gatekeeper is running. However, Azure Policy reports the container as 'ConstraintNotProcessed'.

Cause

The issue could be related to the version or compatibility of your Azure Kubernetes Service (AKS).

Solution

To resolve the issue, upgrade the AKS cluster to a newer or supported version that offers additional features, security enhancements, and updates to prevent issues such as this one.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

AKS container CPU and memory limits aren't enforced

Article • 04/12/2024

This article provides a solution to an issue where CPU and memory limits aren't enforced for containers when you use the Azure Policy Add-on for Azure Kubernetes Service (AKS).

Symptoms

When you deploy an application to an AKS cluster, you receive an error like the following sample:

```
"validation.gatekeeper.sh" denied the request: [container-must-have-limits]  
container <name> has no resource/memory limit
```

Cause

CPU and memory limits aren't configured in the container specification.

Resolution

To resolve this issue, ensure that container CPU and memory limits are enforced by following these steps:

1. Run the following command to check the compliance of your cluster:

```
Console
```

```
kubectl describe k8sazurecontainerlimits
```

This command provides information about the current CPU and memory limits that are configured for your containers.

2. Configure the CPU and memory limits for all your existing deployments in the container specification.

Here's an example:

```
YAML
```

```
spec:  
  containers:  
    - name: cache-service  
      image: xasag94215/flask-cache  
      ports:  
        - containerPort: 5000  
          name: rest  
      resources:  
        requests:  
          cpu: 25m  
          memory: 64Mi  
        limits:  
          cpu: 410m  
          memory: 512Mi
```

ⓘ Note

Adjust the CPU and memory values based on your specific requirements.

3. After updating the container specification, trigger a rescan or on-demand scan to evaluate the compliance status. You can use the Azure CLI to perform an on-demand evaluation scan. For more information about starting an on-demand scan, see [On-demand evaluation scan - Azure CLI](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Custom Azure Policy for validating controllers doesn't work

Article • 04/12/2024

This article provides a solution to an issue where a custom policy created to validate controllers doesn't work when you use Azure Policy with an Azure Kubernetes Service (AKS) cluster.

Symptoms

A custom policy created for validating controllers fails with an error.

Cause

The custom policy isn't defined using the correct syntax.

Solution

To resolve this issue, follow these steps to ensure that the custom policy works correctly and meets the necessary requirements:

1. Generate the custom policy definition with the correct syntax by using Visual Studio Code extension for Azure Policy.

Using this tool to generate a custom policy ensures that it can be properly defined and adhere to the required format.

2. Create and validate the custom policy by referencing [Azure Policy for Kubernetes releases support for custom policy](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Deployment Safeguards in Azure Kubernetes Service (AKS)

08/04/2025

Deployment Safeguards in Azure Kubernetes Service (AKS) help enforce Kubernetes best practices by using Azure Policy and Gatekeeper. While they offer valuable protection, a misconfiguration or misunderstanding of their behavior can cause blocked or mutated workloads. This guide helps you troubleshoot common issues when you use Deployment Safeguards in **Warn** or **Enforce** mode.

Frequently asked questions

Why aren't Deployment Safeguards taking effect?

Symptoms:

- You deploy noncompliant resources, but you see no warnings or signs of enforcement.
- The Azure Policy dashboard shows a **Not started** value or an empty compliance status.

Recommended actions:

- Verify that the Azure Policy add-on is enabled on the cluster:

Bash

```
az aks show --resource-group <rg-name> --name <cluster-name>
--query addonProfiles.azurepolicy
```

- Check whether the namespace is excluded:

Bash

```
az aks safeguards show --resource-group <rg-name> --name
<cluster-name>
```

How can I disable Deployment Safeguards?

To disable deployment safeguards entirely, run the following command:

Bash

```
az aks safeguards delete --resource-group <rg-name> --name  
  <cluster-name>
```

Why can I turn on Deployment Safeguards without Azure Policy permissions?

Deployment Safeguards uses Azure Policy as an implementation detail. To turn on Deployment Safeguards on an AKS cluster, you don't have to have the correct permissions to assign or delete Azure Policies. All that is required are permissions to the AKS Contributor role.

Why does my deployment resource get admitted even though it doesn't follow best practices?

Deployment safeguards enforce best practice standards through Azure Policy controls. It has policies that validate against Kubernetes resources. To evaluate and enforce cluster components, Azure Policy extends [Gatekeeper](#). Gatekeeper enforcement also currently operates in a [fail-open model](#). There are no guarantees that Gatekeeper will respond to our networking call. Therefore, we make sure that the validation doesn't run in such cases so that the denial doesn't block your deployments.

Common error scenarios

When configuring or using Deployment Safeguards, you may encounter error messages in the following situations:

Configuration-related errors

- **Resource group locked:** The managed cluster's resource group has a resource lock that prevents modifications required for Deployment Safeguards.
- **Suspended subscription:** The Azure subscription containing the AKS cluster is in a suspended state.
- **Cluster in deleting state:** You cannot configure Deployment Safeguards on a cluster that is currently being deleted.
- **Unsupported Kubernetes version:** The managed cluster is running a Kubernetes version earlier than 1.25, which is not supported by Deployment Safeguards.

Input validation errors

- **Invalid namespace exclusion format:** Excluded namespaces must follow Kubernetes naming conventions. Values like `ns1,ns2` are not valid - use proper Kubernetes regex patterns.
- **Invalid enforcement level:** The enforcement level must be either `Warn` or `Enforce`. Other values will result in a validation error.
- **Malformed configuration parameters:** Other invalid input parameters will trigger specific validation warnings based on the configuration being applied.

Recommended action: Review the specific warning message and correct the configuration issue before retrying the operation.

Additional tips

- All safeguard policies are bundled. They can't be individually toggled.
- Use the [AKS GitHub repo](#) to request new safeguard features.

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Pods are created in the user namespaces

Article • 05/20/2024

This article discusses a scenario in which pods in Microsoft Azure Kubernetes Service (AKS) are created in the user namespaces. This scenario is generally considered to be a security risk.

Symptoms

When you use the Azure Policy Add-on for AKS and try to create pods by adding the `system-cluster-critical` and `system-node-critical` priority class names in the pod manifests, AKS unexpectedly accepts these names and creates the pods in the user namespaces.

Cause

This is expected behavior. The two system-critical priority classes, `system-cluster-critical` and `system-node-critical`, already exist in the system. Creating pods that have these priority class names in the pod manifests is allowed, even in user namespaces.

Solution

No action is required. The ability to specify the `system-cluster-critical` and `system-node-critical` priority class names in user namespaces is intentional. This ability isn't a security risk by design.

If you have concerns about security, consider reviewing other aspects of your cluster's configuration to enhance security measures.

Resources

- [Pod priority and preemption](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Pods fail and restart after you enable Defender Profile for AKS

Article • 08/07/2024

This article discusses how to troubleshoot a scenario in which the `azure-defender-publisher` DaemonSet pods fail and then restart after you enable Defender Profile for Microsoft Azure Kubernetes Service (AKS).

Prerequisites

- [Azure CLI](#)
- [curl ↗](#)

Symptoms

Consider the following scenario:

- You're using the Azure Policy Add-on for AKS.
- You enable Defender Profile for AKS per the [Azure Advisor](#) recommendation.

In this scenario, the `azure-defender-publisher` DaemonSet pods continually restart.

Cause

This problem might be caused by authentication issues or a misconfiguration.

Workaround

Disable the Azure Defender profile by following these steps:

1. Obtain the access token for your account by running the following [az account get-access-token](#) command:

Azure CLI

```
az account get-access-token --query 'accessToken' --output tsv
```

ⓘ Note

Save the access token from the output to use for commands in the following steps. The access token is an encoded string that can exceed 2,000 characters.

2. To determine whether the Azure Defender profile is enabled on your cluster, invoke the following REST API GET method by running the following curl command. Replace the <access-token> placeholder in the Authorization header with the access token from the previous step, and then replace the placeholders in the URI that represent the Azure subscription GUID, resource group name, and cluster name:

Bash

```
curl -X GET \
-H "Authorization: Bearer <access-token>" \
-H "Content-Type: application/json" \
https://management.azure.com/subscriptions/<subscription-
guid>/resourceGroups/<resource-group-
name>/providers/Microsoft.ContainerService/managedClusters/<cluster-
name>?api-version=2020-04-01
```

If the Azure Defender profile is enabled on your cluster, the command returns JSON output that resembles the following snippet:

JSON

```
"securityProfile": {
  "azureDefender": {
    "enabled": true,
    "logAnalyticsWorkspaceResourceId": "/subscriptions/<subscription-
guid>/resourceGroups/<resource-group-
name>/providers/Microsoft.OperationalInsights/workspaces/DefaultWorkspa
ce-<guid>-<region>"
  }
}
```

3. Remove the Azure Defender profile by invoking the following REST API PUT method through the curl command. This command resembles the previous command, except that you specify `-x PUT` instead of `-x GET` for the method request, and you specify a `-d` parameter to pass data to the request. The required placeholder value replacements are identical to the previous command.

Bash

```
curl -X PUT \
-H "Authorization: Bearer <access-token>" \
-H 'Content-Type: application/json' \
-d '{"location": "northeurope", "properties": {"securityProfile":
```

```
{"azureDefender": {"enabled": false }}}}' \
    https://management.azure.com/subscriptions/<subscription-
    guid>/resourceGroups/<resource-group-
    name>/providers/Microsoft.ContainerService/managedClusters/<cluster-
    name>?api-version=2020-04-01
```

If the command runs as expected, the result shows your cluster in the `Updating` state, and the Azure Defender profile is disabled.

JSON

```
{
  "id": "",
  "location": "northeurope",
  "name": "<cluster-name>",
  "type": "Microsoft.ContainerService/ManagedClusters",
  "properties": {
    "provisioningState": "Updating",
    "powerState": {
      "code": "Running"
    },
    ...
  },
  "securityProfile": {
    "azureDefender": {
      "enabled": false
    }
  }
}
```

After you disable the Azure Defender profile, check whether the `azure-defender-publisher` DaemonSet pods continue their previous cycle of failing and restarting.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

"Policy definition not found" during policy assignments in Terraform

Article • 04/10/2024

This article provides a solution to the "Policy definition not found" error that you encounter during policy assignments when you use the Terraform tool.

Symptoms

When you try to run the following policy assignments that are configured in the Main.tf file of the Terraform tool, you receive the "Policy definition not found" error message:

```
resource "azurerm_policy_assignment" "kubernetes" {
    name          = "kubernetes"
    scope         = <scope>
    policy_definition_id = <policy_definition_id>
    description    = "Assignment of the Kubernetes to subscription."
    display_name   = "Kubernetes-custom-initiative-test01"
}
```

Cause

The error occurs if the policy assignment references the policy initiative definition that's provided for the `policy_definition_id` argument. However, the `azurerm_policy_set_definition` module might be delayed or not found before the policy assignment is created.

Solution

In your Terraform configuration, include a 'depends_on' parameter within the policy assignment resource, as shown the following example. This value makes sure that the policy assignment is generated only after the creation of the policy set definition.

```
resource "azurerm_policy_assignment" "kubernetes" {
    name          = "kubernetes"
    scope         = <scope>
    policy_definition_id = <policy_definition_id>
```

```
    description          = "Assignment of the Kubernetes to subscription."
    display_name        = "Kubernetes-custom-initiative-test01"
    depends_on          = [azurerm_policy_definition.policies]
}
```

For more information, see [the depends_on Meta-Argument](#).

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?



Retain privileged mode while adding capabilities

Article • 04/24/2024

This article discusses how to retain privileged mode while you add capabilities when you use the Microsoft Azure Policy Add-on for Azure Kubernetes Service (AKS).

Overview

Node access always requires a certain level of privilege. Default policies flag these access levels. Even if you don't grant full node access, there are policies that detect access to the file system. In most cases, you have to create exceptions or carefully fine-tune policies based on your requirements.

Although it's possible to specify individual capabilities in the pod's security context without marking the pod as privileged, other policies still apply. You can find a list of these policies in [Azure Kubernetes built-in policy definitions](#). Even if the daemonset's container doesn't trigger the "privileged" policy, the container still requires access to the host's file system in order to copy agent files and scripts and so on. Consider the requirements of your container, and review the policy requirements accordingly.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Third-party scanner detects run-time host incident

Article • 08/07/2024

This article helps you troubleshoot a scenario in which a third-party scanner detects a run-time host incident in Microsoft Azure Kubernetes Service (AKS).

Symptoms

You receive an alert that states that a third-party scanner (such as Prisma) detects a run-time host incident in AKS.

Cause

The alert is typically generated by the container file system.

Solution

Determine which particular container caused the alert. Review the container in question to determine whether there are any alarming indicators. To mitigate any security concerns about unauthorized access to your AKS cluster, make sure that the container and its associated components come from trusted sources.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?



General troubleshooting of the Istio service mesh add-on

Article • 03/18/2025

This article discusses general strategies (that use `kubectl`, `istioctl`, and other tools) to troubleshoot issues that are related to the Istio service mesh add-on for Microsoft Azure Kubernetes Service (AKS). This article also provides a list of possible error messages, reasons for error occurrences, and recommendations to resolve these errors.

Prerequisites

- [Azure CLI](#)
- The Kubernetes [kubectl](#) tool, or a similar tool to connect to the cluster
 - Note:** To install kubectl by using [Azure CLI](#), run the `az aks install-cli` command.
- The Istio [istioctl](#) command-line tool
- The Client URL ([cURL](#)) tool

Troubleshooting checklist: Using `kubectl`

The following troubleshooting steps use various `kubectl` commands to help you debug stuck pods or failures in the Istio daemon (Istiod).

Step 1: Get Istiod pod logs

Get the Istiod pod logs by running the following [kubectl logs](#) command:

```
Bash
```

```
kubectl logs --selector app=istiod --namespace aks-istio-system
```

Step 2: Bounce (delete) a pod

You might have a good reason to restart a pod. Because Istiod is a deployment, it's safe to simply delete the pod by running the [kubectl delete](#) command:

```
Bash
```

```
kubectl delete pods <istio-pod> --namespace aks-istio-system
```

The Istio pod is managed by a deployment. It's automatically re-created and redeployed after you delete it directly. Therefore, deleting the pod is an alternative method for restarting the pod.

ⓘ Note

Alternatively, you can restart the deployment directly by running the following [kubectl rollout restart](#) command:

Bash

```
kubectl rollout restart deployment <istiod-asm-revision> --namespace aks-istio-system
```

Step 3: Check the status of resources

If Istiod isn't scheduled, or if the pod isn't responding, you might want to check the status of the deployment and the replica sets. To do this, run the [kubectl get](#) command:

Bash

```
kubectl get <resource-type> [[--selector app=istiod] | [<resource-name>]]
```

The current resource status appears near the end of the output. The output might also display events that are associated with its controller loop.

Step 4: Get custom resource definition types

To view the types of custom resource definitions (CRDs) that Istio uses, run the [kubectl get](#) command:

Bash

```
kubectl get crd | grep istio
```

To list all the resource names that are based on a particular CRD, run the following [kubectl get](#) command:

Bash

```
kubectl get <crd-type> --all-namespaces
```

Step 5: View the list of Istiod pods

To view the list of Istiod pods, run the following `kubectl get` command:

```
Bash
```

```
kubectl get pod --namespace aks-istio-system --output yaml
```

Step 6: Get more information about the Envoy configuration

If you have connectivity issues between pods, get more information about the Envoy configuration by running the following `kubectl exec` command against Envoy's admin port:

```
Bash
```

```
kubectl exec --namespace <pod-namespace> \
  "$(kubectl get pods \
    --namespace <pod-namespace> \
    --selector app=sleep \
    --output jsonpath='{.items[0].metadata.name}')" \
  --container sleep \
  -- curl -s localhost:15000/clusters
```

Step 7: Get the sidecar logs for the source and destination sidecars

Retrieve the sidecar logs for the source and destination sidecars by running the following `kubectl logs` command two times (the first time for the source pod, and the second time for the destination pod):

```
Bash
```

```
kubectl logs <pod-name> --namespace <pod-namespace> --container istio-proxy
```

Troubleshooting checklist: Using `istioctl`

The following troubleshooting steps describe how to collect information and debug your mesh environment by running various `istioctl` commands.

All `istioctl` commands must be run together with the `--istioNamespace aks-istio-system` flag to point to the AKS add-on installation of Istio.

Warning

Some `istioctl` commands send requests to all sidecars.

① Note

Before you begin, notice that most `istioctl` commands require you to know the control plane revision. You can get this information from the suffix of either the Istiod deployments or the pods, or you can run the following [istioctl tag list](#) command:

```
istioctl tag list
```

Step 1: Make sure that Istio is installed correctly

To verify that you have a correct Istio add-on installation, run the following [istioctl verify-install](#) command:

Bash

```
istioctl verify-install --istioNamespace aks-istio-system --revision <tag>
```

Step 2: Analyze namespaces

To analyze all namespaces, or to analyze a specific namespace, run the following [istioctl analyze](#) command:

Bash

```
istioctl analyze --istioNamespace aks-istio-system \
--revision <tag> \
[--all-namespaces | --namespace <namespace-name>] \
[--failure-threshold {Info | Warning | Error}]
```

Step 3: Get the proxy status

To retrieve the proxy status, run the following [istioctl proxy-status](#) command:

Bash

```
istioctl proxy-status pod/<pod-name> \
--istioNamespace aks-istio-system \
--revision <tag> \
--namespace <pod-namespace>
```

Step 4: Download the proxy configuration

To download the proxy configuration, run the following [istioctl proxy-config all](#) command:

Bash

```
istioctl proxy-config all <pod-name> \
--istioNamespace aks-istio-system \
--namespace <pod-namespace> \
--output json
```

ⓘ Note

Instead of using the `all` variant of the `istioctl proxy-config` command, you can use one of the following variants:

- [istioctl proxy-config ecds](#) ↗ (extension configuration discovery service)
- [istioctl proxy-config bootstrap](#) ↗
- [istioctl proxy-config cluster](#) ↗
- [istioctl proxy-config endpoint](#) ↗
- [istioctl proxy-config listener](#) ↗
- [istioctl proxy-config log](#) ↗
- [istioctl proxy-config route](#) ↗
- [istioctl proxy-config secret](#) ↗

Step 5: Check the injection status

To check the injection status of the resource, run the following `istioctl experimental check-inject` command:

Bash

```
istioctl experimental check-inject --istioNamespace aks-istio-system \
--namespace <pod-namespace> \
--labels <label-selector> | <pod-name> | deployment/<deployment-name>
```

Step 6: Get a full bug report

A full bug report contains the most detailed information. However, running this report can be time-consuming on a large cluster because it includes all pods. You can limit the bug report to certain namespaces. You can also limit the report to certain deployments, pods, or label selectors.

To retrieve a bug report, run the following [istioctl bug-report](#) ↗ command:

Bash

```
istioctl bug-report --istioNamespace aks-istio-system \
    [--include <namespace-1>[, <namespace-2>[, ...]]]
```

Troubleshooting checklist: Miscellaneous issues

Step 1: Fix resource usage issues

If you encounter high memory consumption in Envoy, double-check your Envoy settings for [statistics data collection](#). If you're [customizing Istio metrics](#) through [MeshConfig](#), remember that certain metrics can have [high cardinality](#) and, therefore, create a higher memory footprint. Other fields in MeshConfig, such as concurrency, affect CPU usage and should be configured carefully.

By default, Istio adds information about all services that are in the cluster to every Envoy configuration. The [sidecar](#) can limit the scope of this addition to workloads that are within specific namespaces only. For more information, see [Watch out for this Istio proxy sidecar memory pitfall](#).

For example, the following `Sidecar` definition in the `aks-istio-system` namespace restricts the Envoy configuration for all proxies across the mesh to `aks-istio-system` and other workloads within the same namespace as that specific application:

YAML

```
apiVersion: networking.istio.io/v1alpha3
kind: Sidecar
metadata:
  name: sidecar-restrict-egress
  namespace: aks-istio-system # Needs to be deployed in the root namespace.
spec:
  egress:
    - hosts:
      - "./*"
      - "aks-istio-system/*"
```

You can also try to use the Istio [discoverySelectors](#) option in [MeshConfig](#). The `discoverySelectors` option contains an array of Kubernetes selectors, and it can restrict Istiod's awareness to specific namespaces (as opposed to all namespaces in the cluster). For more information, see [Use discovery selectors to configure namespaces for your Istio service mesh](#).

Step 2: Fix traffic and security misconfiguration issues

To address common traffic management and security misconfiguration issues that Istio users frequently encounter, see [Traffic management problems](#) and [Security problems](#) on the Istio website.

For links to discussion about other issues, such as sidecar injection, observability, and upgrades, see [Common problems](#) on the Istio documentation site.

Step 3: Avoid CoreDNS overload

Issues that relate to CoreDNS overload might require you to change certain Istio DNS settings, such as the `dnsRefreshRate` field in the Istio MeshConfig definition.

Step 4: Fix pod and sidecar race conditions

If your application pod starts before the Envoy sidecar starts, the application might become unresponsive, or it might restart. For instructions about how to avoid this problem, see [Pod or containers start with network issues if istio-proxy is not ready](#). Specifically, setting the `holdApplicationUntilProxyStarts` MeshConfig field under `defaultConfig` to `true` can help prevent these race conditions.

Step 5: Configure a Service Entry when using an HTTP proxy for outbound traffic

If your cluster uses an HTTP proxy for outbound internet access, you'll have to configure a Service Entry. For more information, see [HTTP proxy support in Azure Kubernetes Service](#).

Step 6: Enable Envoy access logging

Enabling Envoy [access logging](#) helps identify and pinpoint issues in the gateways and sidecar proxies. For more information about logging and telemetry collection for the Istio add-on, see the documentation on [mesh configuration](#), [Telemetry API](#), and [Istio metrics collection](#).

Error messages

The following table contains a list of possible error messages (for deploying the add-on, enabling ingress gateways, and performing upgrades), the reason why an error occurred, and recommendations for resolving the error.

 [Expand table](#)

Error	Reason	Recommendations
Azure service mesh is not supported in this region	The feature isn't available in the region during preview (it's available in the public cloud but not the sovereign cloud).	Refer to public documentation about the feature on supported regions.
Missing service mesh mode: {}	You didn't set the mode property in the service mesh profile of the managed cluster request.	In the ServiceMeshProfile field of the <code>managedCluster</code> API request, set the <code>mode</code> property to Istio .
Invalid istio ingress mode: {}	You set an invalid value for the ingress mode when adding ingress within the service mesh profile.	Set the ingress mode in the API request to either External or Internal .
Too many ingresses for type: {}. Only {} ingress gateway are allowed	You tried to create too many ingresses on the cluster.	Create, at most, one external ingress and one internal ingress on the cluster.
Istio profile is missing even though Service Mesh mode is Istio	You enabled the Istio add-on without providing the Istio profile.	When you enable the Istio add-on, specify component-specific (ingress gateway, plug-in CA) information for the Istio profile and

Error	Reason	Recommendations
		the particular revision.
Istio based Azure service mesh is incompatible with feature %s	You tried to use another extension, add-on, or feature that's currently incompatible with the Istio add-on (for example, Open Service Mesh).	Before you enable the Istio add-on, disable the other feature first and clean up all corresponding resources.
ServiceMeshProfile is missing required parameters: %s for plugin certificate authority	You didn't provide all the required parameters for plug-in CA.	Provide all required parameters for the plug-in certificate authority (CA) feature (for more information, see Set up Istio-based service mesh add-on with plug-in CA certificates).
AzureKeyvaultSecretsProvider addon is required for Azure Service Mesh plugin certificate authority feature	You didn't enable the AKS Secrets-Store CSI Driver add-on before you used the plug-in CA.	Set up Azure Key Vault before you use the plug-in CA feature.
'KeyVaultId': '%s' is not a valid Azure keyvault resource identifier. Please make sure that the format matches '/subscriptions//resourceGroups//providers/Microsoft.KeyVault/vaults/'	You used an invalid AKS resource ID.	See the format that's mentioned in the error message to set a valid Azure Key Vault ID for the plug-in CA feature.
Kubernetes version is missing in orchestrator profile	Your request is missing the Kubernetes version. Therefore, it	Make sure that you provide the Kubernetes version in Istio add-on upgrade operations.

Error	Reason	Recommendations
	can't do a version compatibility check.	
Service mesh revision %s is not compatible with cluster version %. To find information about mesh-cluster compatibility, use 'az aks mesh get-upgrades'	You tried to enable an Istio add-on revision that's incompatible with the current Kubernetes cluster version.	Use the az aks mesh get-upgrades Azure CLI command to learn which Istio add-on revisions are available for the current cluster.
Kubernetes version %s not supported. Please upgrade to a supported cluster version first. To find compatibility information, use 'az aks mesh get-upgrades'	You're using an unsupported Kubernetes version.	Upgrade to a supported Kubernetes version.
ServiceMeshProfile revision field must not be empty	You tried to upgrade the Istio add-on without specifying a revision.	Specify the revision and all other parameters (for more information, see Minor revision upgrade).
Request exceeds maximum allowed number of revisions (%d)	You tried to do an upgrade operation even though there are already (%d) revisions installed.	Complete or roll back the upgrade operation before you upgrade to another revision.
Mesh upgrade is in progress. Please complete or roll back the current upgrade before attempting to retrieve versioning and compatibility information	You tried to access revisioning and compatibility information before completing or rolling back the current	Complete or roll back the current upgrade operation before you retrieve revisioning and compatibility information.

Error	Reason	Recommendations
	upgrade operation.	

References

- For general tips about Istio debugging, see [Istio diagnostic tools](#)
- [Istio service mesh add-on MeshConfig troubleshooting](#)
- [Istio service mesh add-on ingress gateway troubleshooting](#)
- [Istio service mesh add-on minor revision upgrade troubleshooting](#)
- [Istio service mesh add-on plug-in CA certificate troubleshooting](#)

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes  No

[Provide product feedback](#)

Istio service mesh add-on ingress gateway troubleshooting

Article • 03/18/2025

This article discusses how to troubleshoot [ingress gateway](#) issues on the Istio service mesh add-on for Azure Kubernetes Service (AKS). The Istio ingress gateway is an [Envoy](#)-based reverse proxy that you can use to route incoming traffic to workloads in the mesh.

For the Istio-based service mesh add-on, we offer the following ingress gateway options:

- An internal ingress gateway that uses a private IP address.
- An external ingress gateway that uses a publicly accessible IP address.

The add-on deploys Istio ingress gateway pods and deployments per revision. If you're doing a [canary upgrade](#) and have two control plane revisions installed in your cluster, then you might have to troubleshoot multiple ingress gateway pods across both revisions.

Troubleshooting checklist

Step 1: Make sure no firewall or NSG rules block the ingress gateway

Verify that you don't have firewall or [Network Security Group \(NSG\) rules](#) that block traffic to the ingress gateway. You have to explicitly add a Destination Network Address Translation (DNAT) rule to [allow inbound traffic](#) through Azure Firewall to the ingress gateway.

Step 2: Configure gateways, virtual services, and destination rules correctly

When you configure gateways, virtual services, and destination rules for traffic routing through the ingress gateway, follow these steps:

1. Make sure that the ingress gateway selector in the gateway resource is set to one of the following text values if you're using an external or internal gateway,

respectively:

- `istio: aks-istio-ingressgateway-external`
- `istio: aks-istio-ingressgateway-internal`

2. Make sure that the ports are set correctly in gateways and virtual services. For the gateway, the port should be set to `80` for `http` or `443` for `https`. For the virtual service, the port should be set to the port that the corresponding service for the application is listening on.
3. Verify that the service is exposed within the `hosts` specification for both the gateway and the virtual service. If you experience issues that are related to the `Host` header in the requests, try adding to the allowlist all hosts that contain an asterisk wildcard ("*"), such as in this [example gateway configuration](#). However, we recommend that you don't amend the allowlist as a production practice. Also, the `hosts` specification should be [configured explicitly](#).

Step 3: Fix the health of the ingress gateway pod

If the ingress gateway pod crashes or doesn't appear in the ready state, verify that the Istio daemon (`istiod`) control plane pod is in the ready state. The ingress gateway depends on having the `istiod` release be ready.

If the `istiod` pod doesn't appear in the ready state, make sure that the Istio custom resource definitions (CRDs) and the `base` Helm chart is installed correctly. To do this, run the following command:

```
Bash
```

```
helm ls --all --all-namespaces
```

You might see a broader error in which the add-on installation isn't configured specifically to the ingress gateway.

If the `istiod` pod is healthy, but the ingress gateway pods aren't responding, inspect the following ingress gateway resources in the `aks-istio-ingress` namespace to collect more information:

- Helm release
- Deployment
- Service

Additionally, you can find more information about gateway and sidecar debugging in [General Istio service mesh add-on troubleshooting](#).

Step 4: Configure resource utilization

High resource utilization occurs when the default min/max replica settings for Istiod and the gateways aren't sufficient. In this case, change [horizontal pod autoscaling](#) configurations.

Step 5: Troubleshoot the secure ingress gateway

When an [external ingress gateway is configured to expose a secure HTTPS service using simple or mutual TLS](#), follow these troubleshooting steps:

1. Verify that the values of the `INGRESS_HOST_EXTERNAL` and `SECURE_INGRESS_PORT_EXTERNAL` environment variables are valid based on the output of the following command:

```
Bash
```

```
kubectl -n aks-istio-ingress get service aks-istio-ingressgateway-  
external
```

2. Check for error messages in the gateway controller's logs:

```
Bash
```

```
kubectl logs -n aks-istio-ingress <gateway-service-pod>
```

3. Verify that the secrets are created in the `aks-istio-ingress` namespace:

```
Bash
```

```
kubectl -n aks-istio-ingress get secrets
```

For the example in [Secure ingress gateway for Istio service mesh add-on for Azure Kubernetes Service](#), the `productpage-credential` secret should be listed.

After you enable the Azure Key Vault secrets provider add-on, you have to grant access for the user-assigned managed identity of the add-on to the Azure Key Vault. Incorrectly setting up access to Azure Key Vault will prevent the `productpage-credential` secret from being created.

After you create the `SecretProviderClass` resource, to ensure secrets sync from Azure Key Vault to the cluster, ensure the sample pod `secrets-store-sync-productpage` that references this resource is successfully deployed.

Step 6: Customize ingress gateway service settings

The add-on also supports [customizing the Kubernetes service for the Istio ingress gateway](#) for certain annotations and the `.spec.externalTrafficPolicy` setting. In certain cases, changing `.spec.externalTrafficPolicy` to `Local` can assist with troubleshooting connectivity and networking issues, as it preserves the client source IP for the incoming request at the ingress gateway.

ⓘ Note

Changing `.spec.externalTrafficPolicy` to `Local` might cause imbalanced traffic spreading. Before applying this change, we recommend reading the Kubernetes documentation about [Preserving the client source IP](#) to understand the tradeoffs between the different `externalTrafficPolicy` settings.

References

- [Istio add-on ingress enablement and configuration](#)
- [Ingress control](#)
- [Traffic management concepts for gateways](#)
- [Istio gateway configuration](#)
- [Istio virtual service configuration](#)
- [Traffic management: Ingress gateway tasks \(TLS termination, secure gateways, and so on\)](#)
- [Istio security best-practices for gateways](#)
- [Traffic management and gateway pitfalls](#)
- [Istio and Kubernetes ingress](#)
- [Kubernetes Gateway API and Istio Ingress](#)

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?



[Provide product feedback](#)

Istio service mesh add-on egress gateway troubleshooting

Article • 05/29/2025

This article discusses how to troubleshoot egress gateway issues on the Istio service mesh add-on for Azure Kubernetes Service (AKS).

Overview

The Istio add-on egress gateway is an Envoy-based proxy that can be used to route outbound traffic from applications in the mesh. The Istio egress gateway is a `ClusterIP` type service and thus isn't exposed externally.

The Istio add-on egress gateway takes a hard dependency on the [Static Egress Gateway feature](#). You must enable the Static Egress Gateway feature on your cluster before enabling an Istio add-on egress gateway.

You can create multiple Istio add-on egress gateways across different namespaces with a `Deployment` or `Service` `name` of your choice, with a max of `500` egress gateways per cluster.

Before troubleshooting

Before proceeding, take the following actions:

- Install Azure CLI `aks-preview` version `14.0.0b2` or later to enable an Istio add-on egress gateway.
- Enable the [Static Egress Gateway feature](#) on your cluster, create an agent pool of mode `gateway`, and configure a `StaticGatewayConfiguration` custom resource.

Networking and firewall errors troubleshooting

Step 1: Make sure no firewall or outbound traffic rules block egress traffic

If you use Azure Firewall, Network Security Group (NSG) rules, or other outbound traffic restrictions, ensure that the IP ranges from the `egressIpPrefix` for the Istio add-on egress gateway `StaticGatewayConfigurations` are allowlisted for egress communication.

Step 2: Verify cluster Container Networking Interface (CNI) plugin

Because Static Egress Gateway is currently not supported on [Azure CNI Pod Subnet](#) clusters, the Istio add-on egress gateway can't be used on Azure CNI Pod Subnet clusters either.

Egress gateway provisioning issues troubleshooting

Step 1: Verify that the Istio egress gateway deployment and service are provisioned

Each Istio egress gateway should have a respective deployment and service provisioned. Verify that the Istio egress gateway deployment and service are provisioned by running the following commands:

Bash

```
kubectl get deployment $ISTIO_EGRESS_DEPLOYMENT_NAME -n $ISTIO_EGRESS_NAMESPACE
```

Bash

```
kubectl get service $ISTIO_EGRESS_NAME -n $ISTIO_EGRESS_NAMESPACE
```

The name of the deployment is in this format: `<istio-egress-gateway-name>-<asm-revision>`. For example, if the name of the egress gateway is `aks-istio-egressgateway` and the Istio add-on revision is `asm-1-24`, the name of the deployment is `aks-istio-egressgateway-asm-1-24`.

You should see a service of type `ClusterIP` for the Istio egress gateway with a service VIP assigned. The name of the service is the same as the name of the Istio egress gateway, without any `revision` appended, such as `aks-istio-egressgateway`.

Step 2: Make sure admission controllers don't block Istio egress provisioning

Ensure that self-managed mutating and validating webhooks don't block provisioning of the Istio egress gateway resources. Because the Istio egress gateway can be deployed in user-managed namespaces, [AKS admissions enforcer](#) can't prevent custom admission controllers from affecting Istio egress gateway resources.

Step 3: Verify that the Istio add-on egress gateway name is valid

Istio egress gateway names must:

- Be 63 characters or fewer in length.
- Only contain lowercase alphanumeric characters, `.`, and `-`.
- Start and end with a lowercase alphanumeric character.
- Be valid Domain Name System (DNS) names.

The regex for Istio egress name validations is `^[a-z0-9]([-a-z0-9]*[a-z0-9])?(\\. [a-z0-9]([-a-z0-9]*[a-z0-9]))?*$`.

Step 4: Inspect Static Egress Gateway components if Istio egress deployments aren't ready

If Static Egress Gateway components such as the `kube-egress-gateway-cni-manager` crash, or there are other issues with the static egress IP allocation, Istio egress gateway provisioning might fail. In this case, [troubleshoot Static Egress Gateway errors or misconfiguration](#).

Static Egress Gateway errors or misconfiguration troubleshooting

Step 1: Inspect the Istio egress gateway StaticGatewayConfiguration

Ensure that the `StaticGatewayConfiguration` for the Istio add-on egress gateway has a valid configuration and isn't deleted:

1. Check the `gatewayConfigurationName` for that egress gateway to find the name of the `StaticGatewayConfiguration` for an Istio add-on egress gateway:

Bash

```
az aks show -g $RESOURCE_GROUP -n $CLUSTER_NAME -o json | jq  
'.serviceMeshProfile.istio.components.egressGateways' | grep  
$ISTIO_EGRESS_NAME -B1
```

2. Verify that the Istio add-on egress gateway pod `spec` has the `kubernetes.azure.com/static-gateway-configuration` annotation set to the

`gatewayConfigurationName` for that Istio add-on egress gateway:

Bash

```
kubectl get pod $ISTIO_EGRESS_POD_NAME -n $ISTIO_EGRESS_NAMESPACE -o jsonpath={.metadata.annotations."kubernetes\.azure\.com\/static-gateway-configuration"}
```

Step 2: Make sure that an `egressIpPrefix` is provisioned for the `StaticGatewayConfiguration`

If the Istio egress gateway pods are stuck in `ContainerCreating`, the `kube-egress-gateway-cni-manager` pod might prevent the `istio-proxy` container from being created because the `StaticGatewayConfiguration` doesn't have an `egressIpPrefix` assigned to it yet. To verify whether it's assigned an `egressIpPrefix`, check the `status` of the `StaticGatewayConfiguration` for that Istio egress gateway. To view if there are any errors with the `egressIpPrefix` provisioning, run the `kubectl describe` command against the `StaticGatewayConfiguration`.

! Note

It can take up to about five minutes for a Static Egress Gateway `StaticGatewayConfiguration` to be assigned an `egressIpPrefix`.

Bash

```
kubectl get staticgatewayconfiguration $ISTIO_SGC_NAME -n $ISTIO_EGRESS_NAMESPACE -o jsonpath='{.status.egressIpPrefix}'  
kubectl describe staticgatewayconfiguration $ISTIO_SGC_NAME -n $ISTIO_EGRESS_NAMESPACE
```

You can also check the logs of the `kube-egress-gateway-cni-manager` pod running on the node of the failing Istio egress pod. If there are issues with `egressIpPrefix` provisioning or if an IP prefix still isn't assigned after approximately five minutes, you might need to [debug the Static Egress Gateway](#) further.

Step 3: Make sure that the `StaticGatewayConfiguration` references a valid gateway agent pool

Verify that the `spec.gatewayNodepoolName` for the `StaticGatewayConfiguration` for each Istio egress gateway references a valid agent pool of mode `Gateway` on the cluster. If any Istio add-

on egress gateway `StaticGatewayConfiguration` references it via the `spec.gatewayNodepoolName`, you shouldn't delete a `Gateway` agent pool.

Step 4: Try sending an external request from the Istio egress gateway

To validate that requests from the Istio egress gateway are routed correctly via the Static Egress Gateway node pool, you can use the `kubectl debug` command to create a Kubernetes ephemeral container and verify the source IP of requests from the Istio egress pod. Make sure that you temporarily set `outboundTrafficPolicy.mode` to `ALLOW_ANY` so that the egress gateway can access `ifconfig.me`. As a security best practice, we recommend setting `outboundTrafficPolicy.mode` back to `REGISTRY_ONLY` after debugging.

Bash

```
kubectl debug -it --image curlimages/curl $ISTIO_EGRESS_POD_NAME -n $ISTIO_EGRESS_NAMESPACE -- curl ifconfig.me
```

The source IP address returned should match the `egressIpPrefix` of the `StaticGatewayConfiguration` associated with that Istio egress gateway. If the request fails or the source IP address returned doesn't match the `egressIpPrefix`, then try [restarting the Istio egress gateway deployment](#) or debugging potential issues with [Static Egress Gateway](#).

Step 5: Try sending a request from an uninjectected pod to the external service

Another way to identify whether the issue is caused by the Istio add-on egress gateway or the Static Egress Gateway is to send a request directly from an uninjectected pod (outside of the Istio mesh). You can use the [curl sample application](#). Under `spec.template.metadata.annotations`, set the `kubernetes.azure.com/static-gateway-configuration` annotation to the same `gatewayConfigurationName` for the Istio add-on egress gateway.

If the requests from the uninjectected pod fail, try debugging potential issues with [Static Egress Gateway](#). If the requests from the uninjectected pod succeed, verify your [Istio egress gateway configurations](#).

Step 6: Try restarting the Istio egress gateway deployment

For changes to certain `StaticGatewayConfiguration` fields such as `defaultRoute` and `excludeCidrs` to take effect, you must restart the Istio add-on egress gateway pods.

You can bounce the pod by triggering a restart of the egress gateway deployment:

```
Bash
```

```
kubectl rollout restart deployment $ISTIO_EGRESS_DEPLOYMENT_NAME -n  
$ISTIO_EGRESS_NAMESPACE
```

Step 7: Try creating a new StaticGatewayConfiguration for the Istio add-on egress gateway

If the `StaticGatewayConfiguration` for the Istio add-on egress gateway has an error, try to create a new `StaticGatewayConfiguration` custom resource in the same namespace. Then, run the following `az aks mesh enable-egress-gateway` command to update the `gatewayConfigurationName`. We recommend waiting until the newly created `StaticGatewayConfiguration` is assigned an `egressIpPrefix`:

```
Bash
```

```
az aks mesh enable-egress-gateway --resource-group $RESOURCE_GROUP --name  
$CLUSTER_NAME --istio-egressgateway-name $ISTIO_EGRESS_NAME --istio-egressgateway-  
namespace $ISTIO_EGRESS_NAMESPACE --gateway-configuration-name $NEW_ISTIO_SGC_NAME
```

After updating the egress gateway to use the new `StaticGatewayConfiguration`, if no other Istio add-on egress gateway uses it, you should be able to delete the previous `StaticGatewayConfiguration`.

Step 8: Debug the Static Egress Gateway

If errors with egress routing through the Istio add-on egress gateway persist even after verifying that [Istio egress routing is configured correctly](#), an underlying networking or infrastructure issue with the Static Egress Gateway might be present. For more information, see the [Configure Static Egress Gateway in Azure Kubernetes Service \(AKS\)](#).

Istio egress configuration and custom resources troubleshooting

For more information about Istio egress configuration, see [Egress Gateways](#).

 Note

Gateway API is currently unsupported for the Istio add-on egress gateway. You must configure the gateway with Istio custom resources to receive support from Azure for issues relating to the Istio egress gateway.

Step 1: Enable Envoy access logging

You can enable Envoy access logging via the [Istio MeshConfig](#) or [Telemetry API](#) to inspect traffic flowing through the egress gateway.

Step 2: Inspect Istio egress gateway and istiod logs

To inspect the logs in the `istio-proxy` container for the Istio add-on egress gateway, run the following command:

Bash

```
kubectl logs $ISTIO_EGRESS_POD_NAME -n $ISTIO_EGRESS_NAMESPACE
```

If you see the `info Envoy proxy is ready` message in the logs, it indicates that the Istio egress gateway pod can communicate with `istiod` and is ready to serve traffic.

We also recommend inspecting the `istiod` control plane logs for any Envoy `xds` errors related to updating the configuration of the Istio egress gateway.

Step 3: Validate the Istio Gateway configuration

Ensure that the `selector` in the `Gateway` custom resource is properly set. For example, if your `Gateway` object for the Istio egress gateway uses the `istio:` selector, then it must match the value of the `istio` label in the Kubernetes service `spec` for that egress gateway.

For example, for an egress gateway with the following Kubernetes service `spec`:

YAML

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    meta.helm.sh/release-name: asm-egx-asn-egress-test
    meta.helm.sh/release-namespace: istio-egress-ns
  creationTimestamp: "2025-04-19T21:50:08Z"
  labels:
    app: asm-egress-test
```

```
azureservicemesh/istio.component: egress-gateway  
istio: asm-egress-test
```

The `Gateway` custom resource should be configured as follows:

YAML

```
apiVersion: networking.istio.io/v1  
kind: Gateway  
metadata:  
  name: istio-egressgateway  
spec:  
  selector:  
    istio: asm-egress-test
```

Step 4: Ensure that a `ServiceEntry` custom resource is created and has a DNS resolution enabled

Ensure that you create a `ServiceEntry` custom resource for the specific external service that the egress gateway forwards requests to. Even if the `outboundTrafficPolicy.mode` is set to `ALLOW_ANY`, create a `ServiceEntry` custom resource might be necessary, since the `Gateway`, `VirtualService`, and `DestinationRule` custom resources might reference an external host via a `ServiceEntry` name. Additionally, when configuring a `ServiceEntry` custom resource for an Istio egress gateway, you must set the `spec.resolution` to `DNS`.

Step 5: Verify the Kubernetes secret namespace for egress gateway mTLS origination

If you try to configure the Istio egress gateway to perform mutual TLS (mTLS) origination, ensure that the Kubernetes secret object is located in the same namespace where the egress gateway is deployed.

Step 6: Ensure that applications send plaintext HTTP requests for egress gateway TLS origination and authorization policies

To originate TLS and apply authorization policies at the egress gateway, workloads in the mesh must send HTTP requests. The sidecar proxies can then use mTLS when forwarding requests to the egress gateway. The egress gateway will terminate the mTLS connection and originate a TLS/HTTPS connection to the destination host.

[Third-party information disclaimer](#)

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Istio service mesh add-on MeshConfig troubleshooting

Article • 06/19/2024

This article discusses how to troubleshoot issues that occur when you [use MeshConfig to configure the Istio service mesh add-on](#) for Microsoft Azure Kubernetes Service (AKS).

Shared ConfigMap configuration

The Istio add-on [MeshConfig](#) enables you to configure certain mesh-wide settings. To do this, you create a local ConfigMap in the `aks-istio-system` namespace. Then, the Istio control plane merges this ConfigMap with the default ConfigMap. (If a conflict exists between settings, the default settings take precedence.) This approach is called a shared ConfigMap configuration.

Create a ConfigMap that's named `istio-shared-configmap-<asm-revision>` in the `aks-istio-system` namespace. For instance, if you use revision `asm-1-18`, you should name the ConfigMap, `istio-shared-configmap-asm-1-18`. Then, you provide the mesh configuration within the `mesh` field of the `data` section, as shown in the following ConfigMap YAML file:

YAML

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: istio-shared-configmap-asm-1-18
  namespace: aks-istio-system
data:
  mesh: |-  
    accessLogFile: /dev/stdout
    defaultConfig:
      holdApplicationUntilProxyStarts: true
```

The values within the `defaultConfig` field are the mesh-wide settings for the Envoy sidecar.

Canary upgrades

Before you start a canary upgrade, follow the [mesh configuration and upgrade guidance](#) to create a second shared ConfigMap for the new control plane revision in the `aks-`

`istio-system` namespace.

If a new ConfigMap wasn't created before you start the upgrade, any features that are configured by the shared ConfigMap won't be accessible. To fix this problem, create the missing ConfigMap for the corresponding revision, and copy over relevant fields from the previous shared ConfigMap.

For more information, see [Upgrade Istio-based service mesh add-on for Azure Kubernetes Service](#) and [Istio service mesh add-on minor revision upgrade troubleshooting](#).

Allowed, supported, and disallowed values

The Istio add-on designates MeshConfig fields as allowed and `supported`, allowed but `unsupported`, and `disallowed`. To learn about allowed and supported MeshConfig fields for the add-on, and see an overview of the different support tiers, see [Configure Istio-based service mesh add-on for Azure Kubernetes Service](#). If fields that are mentioned in the [upstream Istio documentation](#) don't appear in the allowlist for the add-on, these fields are disallowed.

Troubleshooting checklist

Step 1: Make sure that you're editing the correct ConfigMap

- Make sure that you're configuring the shared ConfigMap (for example, `istio-shared-configmap-asm-1-17`) and not editing the default ConfigMap (for example, `istio-asm-1-17`).
- Make sure that the shared ConfigMap points to the correct revision in its title.

Step 2: Remove tab indents from the MeshConfig definition within the shared ConfigMap

In the MeshConfig definition within the shared ConfigMap (under `data.mesh`), make sure that you use spaces instead of tabs. Remove any tab characters that you find.

Step 3: Make sure that MeshConfig fields are valid

If fields are unrecognized or aren't included in the [MeshConfig allowlist](#), updates to the MeshConfig are rejected. Check whether the desired MeshConfig fields are allowed, and make sure that the fields are spelled correctly.

Step 4: Avoid CoreDNS overload

Issues that relate to CoreDNS overload might require you to change certain Istio DNS settings, such as the `dnsRefreshRate` field in the Istio MeshConfig definition.

Step 5: Fix memory consumption issues

If you experience high memory consumption in Envoy, double-check your Envoy settings for [statistics data collection](#). If you're [customizing Istio metrics](#) through the [MeshConfig](#), remember that certain metrics can have [high cardinality](#) and, therefore, cause a higher memory footprint.

We recommend that you use the `discoverySelectors` field in the MeshConfig definition to reduce memory consumption for Istiod and Envoy. For more information, see [General Istio service mesh add-on troubleshooting](#).

Step 6: Free CPU cores

If all CPU cores are in use, the `concurrency` field in the MeshConfig definition might be misconfigured. If this field is set to zero, Envoy uses all the CPU cores. In this situation, remove `concurrency` from the MeshConfig definition. If the `concurrency` field isn't configured, then CPU requests and limits determine the number of CPU cores that are used instead.

Step 7: Fix pod and sidecar race conditions

If your application pod starts before the Envoy sidecar starts, the application might become unresponsive or it might restart. For instructions about how to avoid this problem, see [Pod or containers start with network issues if istio-proxy is not ready](#). Specifically, you can set the `holdApplicationUntilProxyStarts` MeshConfig field under `defaultConfig` to `true` to help prevent these race conditions.

References

- [Configure Istio-based service mesh add-on for Azure Kubernetes Service](#)

- General Istio service mesh add-on troubleshooting
- Istio service mesh add-on ingress gateway troubleshooting
- Istio service mesh add-on minor revision upgrade troubleshooting
- Istio service mesh add-on plug-in CA certificate troubleshooting

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Istio service mesh add-on minor revision upgrade troubleshooting

Article • 02/07/2025

This article discusses troubleshooting scenarios and restrictions in the minor revision upgrade and rollback processes for the Istio service mesh add-on in Microsoft Azure Kubernetes Service (AKS).

ⓘ Note

Istio uses the term "revisions" to implement the canary upgrade process and distinguish between versions. Each revision designation (written as $x-y$) corresponds to a major.minor version designation ($x.y$). You can control your control plane revision, but you can't control the specific patch version within a revision band.

Prerequisites

- [Azure CLI](#).
- The Kubernetes [kubectl](#) tool, or a similar tool to connect to the cluster. To install kubectl by using [Azure CLI](#), run the `az aks install-cli` command.

Troubleshooting matrix

The following table lists various problems and the different scenarios and solutions for those problems.

[+] [Expand table](#)

Scenario	Problem	Solution
Data plane workloads are dropped from the mesh.	Data plane and control plane revisions didn't correspond before you completed or rolled back an upgrade.	<p>Follow these steps:</p> <ol style="list-style-type: none">1. Relabel namespaces that contain workloads by specifying the revision that's expected to exist after the upgrade completion or rollback: <pre>kubectl label namespace default istio.io/rev=asm-x-y --overwrite</pre>

Scenario	Problem	Solution
		<p>2. Restart the corresponding workload deployments using the kubectl rollout restart command to trigger sidecar reinjection of the correct revision:</p> <pre>kubectl rollout restart deployment <deployment name></pre> <p>3. Verify that the sidecar images exist using the kubectl get command:</p> <pre>kubectl get pods --namespace <namespace> --output yaml grep mcr.microsoft.com/oss/istio/proxyv2:</pre>
Control plane pods are in the pending state.	The pods lack capacity.	Verify the state of the pods by running the kubectl describe command. If capacity is the problem, you can scale up your cluster to add another node. For more information, see Manually scale the node count in an Azure Kubernetes Service (AKS) cluster .
The az aks mesh get-upgrades command returns no available upgrades.	The next Istio revision might be incompatible with the current AKS cluster version.	You can use the az aks mesh get-revisions command to discover whether newer Istio revisions exist. The output includes a list of compatible cluster versions for each Istio revision. Therefore, you can determine whether a cluster upgrade is necessary. If <i>both</i> mesh and cluster are no longer supported, upgrade the cluster version first, and then the mesh revision. To recover from this scenario, a cluster upgrade is permitted even if it's incompatible with the mesh revision.

ⓘ Note

To avoid unintended behavior and broken functionality, and also make sure that you're receiving updates for security vulnerabilities, we strongly recommend that you upgrade to a supported and up-to-date [AKS version](#) and Istio add-on revision. Remember that the add-on revision should also be within the supported Kubernetes version range for the given AKS cluster. As highlighted in the [Minor revision upgrade](#) section of the Istio upgrade article, you can run the `az aks mesh get-revisions` and `az aks mesh get-upgrades` commands to learn about available add-on revisions, upgrades, and compatibility information.

Restrictions

- A downgrade to an older revision (outside the canary rollback process) isn't allowed.
- Available upgrades for a revision depend on whether it's currently supported. For example, if `n` is the currently installed revision and `n+2` is the latest revision:
 - If `n` is supported, you may upgrade to the next revision `n+1` or directly to the newest revision `n+2`.
 - If both `n` and `n+1` (next consecutive) are unsupported, the only available upgrade is `n+2` (next supported).
 - If `n` has been unsupported for a while, it's possible both of the next two consecutive revisions are unsupported. In this case, the only available upgrade is the lowest supported revision.
- The Istio `sidecar.istio.io/inject` label doesn't enable sidecar injection for the Istio add-on. You must use the `istio.io/rev` label when you label and relabel your namespaces during the canary upgrade.
- Labeling must occur on a namespace level instead of on a per-deployment level. If you want to be able to roll over pods individually, you can choose to restart individual deployments instead of using pod labeling.
- If you're using the Istio add-on Shared MeshConfig, you have to copy or transfer MeshConfig settings to the new ConfigMap before you do a canary upgrade. For more information, see [Mesh configuration and upgrades](#).
- The Istio add-on deploys Istio ingress gateway pods and deployments per revision. If you're doing a [canary upgrade](#) and have two control plane revisions installed in your cluster, you might have to troubleshoot multiple ingress gateway pods across both revisions.

References

- [Safely upgrade the Istio control plane with revisions and tags](#)
- [General Istio service mesh add-on troubleshooting](#)
- [Istio service mesh add-on MeshConfig troubleshooting](#)
- [Istio service mesh add-on ingress gateway troubleshooting](#)
- [Istio service mesh add-on plug-in CA certificate troubleshooting](#)

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

Istio service mesh add-on plug-in CA certificate troubleshooting

Article • 04/01/2025

This article discusses common troubleshooting issues with the Istio add-on plug-in certificate authority (CA) certificates feature, and it offers solutions to fix these issues. The article also reviews the general process of setting up plug-in CA certificates for the service mesh add-on.

ⓘ Note

This article assumes that Istio revision `asm-1-21` is deployed on the cluster.

Prerequisites

- [Azure CLI](#).
- The Kubernetes [kubectl](#) tool, or a similar tool, to connect to the cluster. To install kubectl by using Azure CLI, run the `az aks install-cli` command.
- The following Linux-style standard shell tools:
 - `grep`
 - `sort`
 - `tail`
 - `awk`
 - `xargs`
- The [jq](#) tool for querying JSON data.

General setup process

- Before you enable the Istio add-on to use the plug-in CA certificates feature, you have to enable the [Azure Key Vault provider for Secrets Store add-on](#) on the cluster. Make sure that the Azure Key Vault and the cluster are on the same Azure tenant.
- After the Azure Key Vault secrets provider add-on is enabled, you have to set up access to the Azure Key Vault for the [user-assigned managed identity](#) that the add-on creates.

- After you grant permission for the user-assigned managed identity to access the Azure Key Vault, you can use the plug-in CA certificates feature together with the Istio add-on. For more information, see the [Enable the Istio add-on to use a plug-in CA certificate](#) section.
- For the cluster to auto-detect changes in the Azure Key Vault secrets, you have to enable [auto-rotation](#) for the Azure Key Vault secrets provider add-on.
- Changes to the root and intermediate certificates are applied automatically.

Enable the Istio add-on to use a plug-in CA certificate

The Istio add-on [plug-in CA certificates](#) feature allows you to configure plug-in root and intermediate certificates for the mesh. To provide plug-in certificate information when you enable the add-on, specify the following parameters for the [az aks mesh enable](#) command in Azure CLI.

[] [Expand table](#)

Parameter	Description
<code>--key-vault-id <resource-id></code>	The Azure Key Vault resource ID. This resource is expected to be in the same tenant as the managed cluster. This resource ID must be in the Azure Resource Manager template (ARM template) resource ID format .
<code>--root-cert-object-name <root-cert-object-name></code>	The root certificate object name in the Azure Key Vault.
<code>--ca-cert-object-name <inter-cert-object-name></code>	The intermediate certificate object name in the Azure Key Vault.
<code>--ca-key-object-name <inter-key-object-name></code>	The intermediate certificate private key object name in the Azure Key Vault.
<code>--cert-chain-object-name <cert-chain-object-name></code>	The certificate chain object name in the Azure Key Vault.

If you want to use the plug-in CA certificates feature, you must specify all five parameters. All Azure Key Vault objects are expected to be of the type [Secret](#).

For more information, see [Plug in CA certificates for Istio-based service mesh add-on on Azure Kubernetes Service](#).

Deployed resources

As part of the add-on deployment for the plug-in certificates feature, the following resources are deployed onto the cluster:

- The `istio-spc-asm-1-21` [SecretProviderClass object](#) is created in the `aks-istio-system` namespace at the time of the add-on deployment. This resource contains Azure-specific parameters for the Secrets Store Container Storage Interface (CSI) driver:

Bash

```
kubectl get secretproviderclass --namespace aks-istio-system
```

Output

NAME	AGE
<code>istio-spc-asm-1-21</code>	14h

- The `istio-ca-root-cert` configmap is created in the `aks-istio-system` namespace and all user-managed namespaces. This configmap contains the root certificate that the certificate authority uses, and it's used by workloads in the namespaces to validate workload-to-workload communication, as follows:

Bash

```
kubectl describe configmap istio-ca-root-cert --namespace aks-istio-system
```

Output

Name:	<code>istio-ca-root-cert</code>
Namespace:	<code>aks-istio-system</code>
Labels:	<code>istio.io/config=true</code>
Annotations:	<code><none></code>

Data

====

`root-cert.pem:`

-----BEGIN CERTIFICATE-----

```
<certificate data>
-----END CERTIFICATE-----
```

Determine certificate type in deployment logs

You can view the `istiod` deployment logs to determine whether you have a self-signed CA certificate or a plug-in CA certificate. To view the logs, run the following command:

Bash

```
kubectl logs deploy/istiod-asm-1-21 --container discovery --namespace aks-istio-system | grep -v validationController
```

Immediately before each certificate log entry is another log entry that describes that kind of certificate. For a self-signed CA certificate, the entry states "No plugged-in cert at `etc/cacerts/ca-key.pem`; self-signed cert is used." For a plug-in certificate, the entry states "Use plugged-in cert at `etc/cacerts/ca-key.pem`." Sample log entries that pertain to the certificates are shown in the following tables.

- Log entries for a self-signed CA certificate

[\[+\] Expand table](#)

Timestamp	Log level	Message
2023-11-20T23:27:36.649019Z	info	Using istiod file format for signing ca files
2023-11-20T23:27:36.649032Z	info	No plugged-in cert at <code>etc/cacerts/ca-key.pem</code> ; self-signed cert is used
2023-11-20T23:27:36.649536Z	info	x509 cert - <certificate-details>
2023-11-20T23:27:36.649552Z	info	Istiod certificates are reloaded
2023-11-20T23:27:36.649613Z	info	spiffe Added 1 certs to trust domain cluster.local in peer cert verifier

- Log entries for a plug-in CA certificate

[\[+\] Expand table](#)

Timestamp	Log level	Message
2023-11-21T00:20:25.808396Z	info	Using istiod file format for signing ca files
2023-11-21T00:20:25.808412Z	info	Use plugged-in cert at etc/cacerts/ca-key.pem
2023-11-21T00:20:25.808731Z	info	x509 cert - <certificate-details>
2023-11-21T00:20:25.808764Z	info	x509 cert - <certificate-details>
2023-11-21T00:20:25.808799Z	info	x509 cert - <certificate-details>
2023-11-21T00:20:25.808803Z	info	Istiod certificates are reloaded
2023-11-21T00:20:25.808873Z	info	spiffe Added 1 certs to trust domain cluster.local in peer cert verifier

The certificate details in a log entry are shown as comma-separated values for the issuer, subject, serial number (SN—a long hexadecimal string), and the beginning and ending timestamp values that define when the certificate is valid.

For a self-signed CA certificate, there's one detail entry. Sample values for this certificate are shown in the following table.

[Expand table](#)

Issuer	Subject	SN	NotBefore	NotAfter
"O=cluster.local"	""	<32-digit-hex-value>	"2023-11-20T23:25:36Z"	"2033-11-17T23:27:36Z"

For a plug-in CA certificate, there are three detail entries. The other two entries are for a root certificate update and a change to the intermediate certificate. Sample values for these entries are shown in the following table.

[Expand table](#)

Issuer	Subject	SN	NotBefore	NotAfter
CN=Intermediate CA - A1,O=Istio,L=cluster-	""	<32-digit-	"2023-11-21T00:18:25Z"	"2033-11-18T00:20:25Z"

Issuer	Subject	SN	NotBefore	NotAfter
A1"		hex-value>		
CN=Root A,O=Istio"	"CN=Intermediate CA - A1,O=Istio,L=cluster-A1"	<40-digit-hex-value>	"2023-11-04T01:40:22Z"	"2033-11-01T01:40:22Z"
CN=Root A,O=Istio"	"CN=Root A,O=Istio"	<40-digit-hex-value>	"2023-11-04T01:38:27Z"	"2033-11-01T01:38:27Z"

Troubleshoot common issues

Issue 1: Access to Azure Key Vault is set up incorrectly

After you enable the Azure Key Vault secrets provider add-on, you have to grant access for the user-assigned managed identity of the add-on to the Azure Key Vault. Setting up access to Azure Key Vault incorrectly causes the add-on installation to stall.

Bash

```
kubectl get pods --namespace aks-istio-system
```

In the list of pods, you can see that the `istiod-asm-1-21` pods are stuck in an `Init:0/2` state.

[\[+\] Expand table](#)

NAME	READY	STATUS	RESTARTS	AGE
istiod-asm-1-21-6fcfd88478-2x95b	0/1	Terminating	0	5m55s
istiod-asm-1-21-6fcfd88478-6x5hh	0/1	Terminating	0	5m40s
istiod-asm-1-21-6fcfd88478-c48f9	0/1	Init:0/2	0	54s
istiod-asm-1-21-6fcfd88478-wl8mw	0/1	Init:0/2	0	39s

To verify the Azure Key Vault access issue, run the `kubectl get pods` command to locate pods that have the `secrets-store-provider-azure` label in the `kube-system` namespace:

Bash

```
kubectl get pods --selector app=secrets-store-provider-azure --namespace kube-system --output name | xargs -I {} kubectl logs --namespace kube-system {}
```

The following sample output shows that a "403 Forbidden" error occurred because you don't have "get" permissions for secrets on the Key Vault:

```
"failed to process mount request" err="failed to get objectType:secret, objectName: <secret-object-name>, objectVersion:: keyvault.BaseClient#GetSecret: Failure responding to request: StatusCode=403 -- Original Error: autorest/azure: Service returned an error. Status=403 Code=\"Forbidden\" Message=\"The user, group or application 'appid=<appid>;oid=<oid>;iss=<iss>' does not have secrets get permission on key vault 'MyAzureKeyVault;location=eastus'. For help resolving this issue, please see https://go.microsoft.com/fwlink/?linkid=2125287\" InnerError={"code\":\"AccessDenied\"}"
```

To fix this problem, set up access to the user-assigned managed identity for the Azure Key Vault secrets provider add-on by obtaining Get and List permissions on Azure Key Vault secrets and reinstalling the Istio add-on. First, get the object ID of the user-assigned managed identity for the Azure Key Vault secrets provider add-on by running the [az aks show](#) command:

Azure CLI

```
OBJECT_ID=$(az aks show --resource-group $RESOURCE_GROUP --name $CLUSTER --query 'addonProfiles.azureKeyvaultSecretsProvider.identity.objectId')
```

To set the access policy, run the following [az keyvault set-policy](#) command by specifying the object ID that you obtained:

Azure CLI

```
az keyvault set-policy --name $AKV_NAME --object-id $OBJECT_ID --secret-permissions get list
```

ⓘ Note

Did you create your Key Vault by using Azure RBAC Authorization for your permission model instead of Vault Access Policy? In this case, see [Provide access to Key Vault keys, certificates, and secrets with an Azure role-based access control](#)

to create permissions for the managed identity. Add an Azure role assignment for [Key Vault Reader](#) for the user-assigned managed identity of the add-on.

Issue 2: Auto-detection of Key Vault secret changes isn't set up

For a cluster to auto-detect changes in the Azure Key Vault secrets, you have to enable auto-rotation for the Azure Key Vault provider add-on. Auto-rotation can detect changes in intermediate and root certificates automatically. For a cluster that enables the Azure Key Vault provider add-on, run the following `az aks show` command to check whether auto-rotation is enabled:

Azure CLI

```
az aks show --resource-group $RESOURCE_GROUP --name $CLUSTER | jq -r  
'.addonProfiles.azureKeyvaultSecretsProvider.config.enableSecretRotation'
```

If the cluster enabled the Azure Key Vault provider add-on, run the following `az aks show` command to determine the rotation poll interval:

Azure CLI

```
az aks show --resource-group $RESOURCE_GROUP --name $CLUSTER | jq -r  
'.addonProfiles.azureKeyvaultSecretsProvider.config.rotationPollInterval'
```

Azure Key Vault secrets are synchronized with the cluster when the poll interval time elapses after the previous synchronization. The default interval value is two minutes.

Issue 3: Certificate values are missing or are configured incorrectly

If secret objects are missing from Azure Key Vault, or if these objects are configured incorrectly, the `istiod-asm-1-21` pods might get stuck in an `Init:0/2` status, delaying the installation of the add-on. To find the underlying cause of this problem, run the following `kubectl describe` command against the `istiod` deployment and view the output:

Bash

```
kubectl describe deploy/istiod-asm-1-21 --namespace aks-istio-system
```

The command displays events that might resemble the following output table. In this example, a missing secret is the cause of the problem.

[+] Expand table

Type	Reason	Age	From	Message
Normal	Scheduled	3m9s	default-scheduler	Successfully assigned aks-istio-system/istiod-asm-1-21-6fcfd88478-hqdjj to aks-userpool-24672518-vmss000000
Warning	FailedMount	66s	kubelet	Unable to attach or mount volumes: unmounted volumes=[cacerts], unattached volumes=[], failed to process volumes=[]: timed out waiting for the condition
Warning	FailedMount	61s (x9 over 3m9s)	kubelet	MountVolume.SetUp failed for volume "cacerts" : rpc error: code = Unknown desc = failed to mount secrets store objects for pod aks-istio-system/istiod-asm-1-21-6fcfd88478-hqdjj, err: rpc error: code = Unknown desc = failed to mount objects, error: failed to get objectType:secret, objectName:test-cert-chain, objectVersion:: keyvault.BaseClient#GetSecret: Failure responding to request: StatusCode=404 -- Original Error: autorest/azure: Service returned an error. Status=404 Code="SecretNotFound" Message="A secret with (name/id) test-cert-chain was not found in this key vault. If you recently deleted this secret you may be able to recover it using the correct recovery command. For help resolving this issue, please see https://go.microsoft.com/fwlink/?linkid=2125182"

Resources

- [General Istio service mesh add-on troubleshooting](#)
- [Istio service mesh add-on MeshConfig troubleshooting](#)
- [Istio service mesh add-on ingress gateway troubleshooting](#)
- [Istio service mesh add-on minor revision upgrade troubleshooting](#)

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?



[Provide product feedback](#)

AKS Cost Analysis add-on issues

07/09/2025

This article discusses how to troubleshoot problems that you might experience when you enable the Microsoft Azure Kubernetes Service (AKS) Cost Analysis add-on during cluster creation or a cluster update.

Prerequisites

- [Azure CLI](#)

Symptoms

After you create or update an AKS cluster, you receive an error message in the following format:

[] [Expand table](#)

Error code	Cause
<code>InvalidDiskCSISettingForCostAnalysis</code>	Cause 1: Azure Disk CSI driver is disabled
<code>InvalidManagedIdentitySettingForCostAnalysis</code>	Cause 2: Managed identity is disabled
<code>CostAnalysisNotEnabledInRegion</code>	Cause 3: The add-on is unavailable in your region
<code>InvalidManagedClusterSKUForFeature</code>	Cause 4: The add-on is unavailable on the free pricing tier
Pod <code>oomKilled</code>	Cause 5: The cost-analysis-agent pod gets the OOMKilled error
Pod <code>Pending</code>	Cause 6: The cost-analysis-agent pod is stuck in the Pending state

Cause 1: Azure Disk CSI driver is disabled

You can't enable the Cost Analysis add-on on a cluster in which the [Azure Disk Container Storage Interface \(CSI\) driver](#) is disabled.

Solution: Update the cluster to enable the Azure Disk CSI driver

Run the `az aks update` command, and specify the `--enable-disk-driver` parameter. This parameter enables the Azure Disk CSI driver in AKS.

First, define the environment variables for your resource group and AKS cluster, using unique values for repeated runs:

Azure CLI

```
export RANDOM_SUFFIX=$(head -c 3 /dev/urandom | xxd -p)
export RESOURCE_GROUP="my-aks-resource-group$RANDOM_SUFFIX"
export AKS_CLUSTER="my-aks-cluster$RANDOM_SUFFIX"
az aks update --resource-group $RESOURCE_GROUP --name $AKS_CLUSTER --enable-disk-driver
```

For more information, see [CSI drivers on AKS](#).

Cause 2: Managed identity is disabled

You can enable the Cost Analysis add-on only on a cluster that has a system-assigned or user-assigned managed identity.

Solution: Update the cluster to enable managed identity

Run the `az aks update` command, and specify the `--enable-managed-identity` parameter:

Azure CLI

```
export RANDOM_SUFFIX=$(head -c 3 /dev/urandom | xxd -p)
export RESOURCE_GROUP="my-aks-resource-group$RANDOM_SUFFIX"
export AKS_CLUSTER="my-aks-cluster$RANDOM_SUFFIX"
az aks update --resource-group $RESOURCE_GROUP --name $AKS_CLUSTER --enable-managed-identity
```

For more information, see [Use a managed identity in AKS](#).

Cause 3: The add-on is unavailable in your region

The Cost Analysis add-on isn't currently enabled in your region.

! Note

The AKS Cost Analysis add-on is currently unavailable in the following regions:

- usnateast
- usnatwest
- usseceast
- ussecwest

Cause 4: The add-on is unavailable on the free pricing tier

You can't enable the Cost Analysis add-on on AKS clusters that are on the free pricing tier.

Solution: Update the cluster to use the Standard or Premium pricing tier

Upgrade the AKS cluster to the Standard or Premium pricing tier. To do this, run the below `az aks update` command that specify the `--tier` parameter. The `--tier` parameter can be set to either `standard` or `premium` (example below shows `standard`):

Azure CLI

```
export RANDOM_SUFFIX=$(head -c 3 /dev/urandom | xxd -p)
export RESOURCE_GROUP="my-aks-resource-group$RANDOM_SUFFIX"
export AKS_CLUSTER="my-aks-cluster$RANDOM_SUFFIX"
az aks update --resource-group $RESOURCE_GROUP --name $AKS_CLUSTER --tier standard
```

For more information, see [Free and Standard pricing tiers for AKS cluster management](#).

Cause 5: The cost-analysis-agent pod gets the OOMKilled error

The current memory limit for the cost-analysis-agent pod is set to 4 GB.

The pod's usage depends on the number of deployed containers, which can be roughly 200 MB + 0.5 MB per container. The current memory limit supports approximately 7000 containers per cluster.

When the pod's usage exceeds the allocated 4 GB limit, large clusters may experience the `OOMKill` error.

Solution: Disable the add-on

Currently, customizing or manually increasing memory limits for the add-on isn't supported. To resolve this issue, disable the add-on.

Cause 6: The cost-analysis-agent pod is stuck in the Pending state

If the pod is stuck in the Pending state with the FailedScheduling error, the nodes in the cluster have exhausted memory capacity.

Solution: Ensure there's sufficient allocatable memory

The current memory request of the cost-analysis-agent pod is set to 500 MB. Ensure that there's sufficient allocatable memory for the pod to be scheduled

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Troubleshoot AKS AI toolchain operator add-on errors

Article • 05/16/2025

This article provides guidance on resolving errors that might occur when you enable the Microsoft Azure Kubernetes Service (AKS) AI Toolchain Operator (KAITO) add-on during cluster creation or update.

Prerequisites

Ensure the following tools are installed and configured. They're used in the following sections.

- [Azure CLI](#)
- [kubectl](#), the Kubernetes command-line client

Symptoms

The KAITO add-on consists of two controllers: the `gpu-provisioner` controller and the `workspace` controller. After enabling the add-on and deploying a KAITO workspace, you might encounter one or more of the following errors in your pod logs:

[Expand table](#)

Error message	Cause
Workspace was not created	Cause 1: Incorrect KAITO custom resource configuration
GPU node was not created	Cause 2: GPU quota limitations
Resource ready condition is not <code>True</code>	Cause 3: Long pull time for model inference images

Cause 1: Incorrect KAITO custom resource configuration

After you enable the add-on and deploy a preset or custom workspace custom resource (CR), the `workspace` controller includes a validation webhook. This webhook blocks common mistakes of setting wrong values in the CR specification.

To resolve this issue, follow these steps:

1. Check your `gpu-provisioner` and `workspace` pod logs.

2. Ensure that any updates to the GPU virtual machine (VM) size meet the requirements of your model size.
3. Once the workspace CR is successfully created, track the deployment progress by running the following commands:

```
Azure CLI
```

```
kubectl get machine -o wide
```

```
Azure CLI
```

```
kubectl get workspace -o wide
```

Cause 2: GPU quota limitations

The `gpu-provisioner` controller might fail to create GPU nodes due to quota limitations in your subscription or region. In this case, you can check the machine CR status (internal CR created by the `workspace` controller) for error messages. The machine CR created by the `workspace` controller has a `kaito.sh/workspace` label key whose value is the workspace's name.

To resolve this issue, use one of the following methods:

- Request an [increase in the subscription quota](#) for the required GPU VM family of your deployment.
- Check the [GPU instance availability](#) in the specific region of your AKS cluster.

If the required GPU VM size is unavailable in your current region, consider switching to a different region or selecting an alternative GPU VM size.

Cause 3: Long pull time for model inference images

If the image access mode is set to private, the model inference image might not be pulled. This issue can occur for images with specified URLs and pull secrets.

Inference images are typically large (30 GB -100 GB), so a longer image pull time is expected. Depending on your AKS cluster's networking setup, the pull process might take up to tens of minutes.

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Fail to pull images from Azure Container Registry to Azure Kubernetes Service cluster

06/05/2025

! Note

Was this article helpful? Your input is important to us. Please use the **Feedback** button on this page to let us know how well this article worked for you or how we can improve it.

When you're using Microsoft Azure Container Registry together with Azure Kubernetes Service (AKS), an authentication mechanism must be established. You can set up the AKS to Container Registry integration by using a few simple Azure CLI or Azure PowerShell commands. This integration assigns the [AcrPull role](#) for the kubelet identity that's associated with the AKS cluster to pull images from a container registry.

In some cases, trying to pull images from a container registry to an AKS cluster fails. This article provides guidance for troubleshooting the most common errors that you encounter when you pull images from a container registry to an AKS cluster.

Before you begin

This article assumes that you have an existing AKS cluster and an existing container registry.

See the following quick starts:

- If you need an AKS cluster, deploy one by using [the Azure CLI](#) or [the Azure portal](#).
- If you need an Azure Container Registry (ACR), create one by using [the Azure CLI](#) or [the Azure portal](#).

You also need Azure CLI version 2.0.59 or a later version to be installed and configured. Run [az version](#) to determine the version. If you have to install or upgrade, see [Install Azure CLI](#).

Symptoms and initial troubleshooting

The Kubernetes pod's **STATUS** is **ImagePullBackOff** or **ErrImagePull**. To get detailed error information, run the following command and check **Events** from the output.

Console

```
kubectl describe pod <podname> -n <namespace>
```

We recommend that you start troubleshooting by checking the [container registry's health](#) and checking whether the container registry is accessible from the AKS cluster.

To check the container registry's health, run the following command:

Azure CLI

```
az acr check-health --name <myregistry> --ignore-errors --yes
```

If a problem is detected, it provides an error code and description. For more information about the errors and possible solutions, see [Health check error reference](#).

 **Note**

If you get Helm-related or Notary-related errors, it doesn't mean that you an issue is affecting Container Registry or AKS. It indicates only that Helm or Notary isn't installed, or that Azure CLI isn't compatible with the current installed version of Helm or Notary, and so on.

To validate whether the container registry is accessible from the AKS cluster, run the following [az aks check-acr](#) command:

Azure CLI

```
az aks check-acr --resource-group <MyResourceGroup> --name <MyManagedCluster> --acr <myacr>.azurecr.io
```

The following sections help you troubleshoot the most common errors that are displayed in **Events** in the output of the `kubectl describe pod` command.

Cause 1: 401 Unauthorized error

Cause 1a: 401 Unauthorized error due to incorrect authorization

An AKS cluster requires an identity. This identity can be either a managed identity or a service principal. If the AKS cluster uses a managed identity, the kubelet identity is used for authenticating with ACR. If the AKS cluster is using as an identity a service principal, the service

principal itself is used for authenticating with ACR. No matter what the identity is, the proper authorization that's used to pull an image from a container registry is necessary. Otherwise, you may get the following "401 Unauthorized" error:

```
Failed to pull image "<acrname>.azurecr.io/<repository:tag>": [rpc error: code = Unknown  
desc = failed to pull and unpack image "<acrname>.azurecr.io/<repository:tag>": failed to  
resolve reference "<acrname>.azurecr.io/<repository:tag>": failed to authorize: failed to  
fetch oauth token: unexpected status: 401 Unauthorized
```

Several solutions can help you resolve this error, subject to the following constraints:

- Solutions [2](#), [3](#), and [5](#) are applicable only to [AKS clusters that use a service principal](#).
- Solutions [1](#), [2](#), [3](#), and [4](#) are applicable for the Azure method of [creating the role assignment at Container Registry level for AKS's identity](#).
- Solutions [5](#) and [6](#) are applicable for the Kubernetes method of [pulling a Kubernetes secret](#).

Solution 1: Make sure AcrPull role assignment is created for identity

The integration between AKS and Container Registry creates an AcrPull role assignment at container registry level for the AKS cluster's kubelet identity. Make sure that the role assignment is created.

To check whether the AcrPull role assignment is created, use one of the following methods:

- Run the following command:

Azure CLI

```
az role assignment list --scope  
/subscriptions/<subscriptionID>/resourceGroups/<resourcegroupname>/providers/  
Microsoft.ContainerRegistry/registries/<acrname> -o table
```

- Check in the Azure portal by selecting [Azure Container Registry > Access control \(IAM\) > Role assignments](#). For more information, see [List Azure role assignments using the Azure portal](#).

Besides the AcrPull role, some [built-in roles](#) and [custom roles](#) can also contain the "[Microsoft.ContainerRegistry/registries/pull/read](#)" action. Check those roles if you've got any of them.

If the AcrPull role assignment isn't created, create it by [configuring Container Registry integration for the AKS cluster](#) with the following command:

Azure CLI

```
az aks update -n <myAKSCluster> -g <myResourceGroup> --attach-acr <acr-resource-id>
```

Solution 2: Make sure service principal isn't expired

Make sure that the secret of the service principal that's associated with the AKS cluster isn't expired. To check the expiration date of your service principal, run the following commands:

Azure CLI

```
SP_ID=$(az aks show --resource-group <myResourceGroup> --name <myAKSCluster> \
--query servicePrincipalProfile.clientId -o tsv)

az ad app credential list --id "SP_ID" --query "[].endDateTime" --output tsv
```

For more information, see [Check the expiration date of your service principal](#).

If the secret is expired, [update the credentials for the AKS cluster](#).

Solution 3: Make sure AcrPull role is assigned to correct service principal

In some cases, the container registry role assignment still refers to the old service principal. For example, when the service principal of the AKS cluster is replaced with a new one. To make sure that the container registry role assignment refers to the correct service principal, follow these steps:

1. To check the service principal that's used by the AKS cluster, run the following command:

Azure CLI

```
az aks show --resource-group <myResourceGroup> \
--name <myAKSCluster> \
--query servicePrincipalProfile.clientId \
--output tsv
```

2. To check the service principal that's referenced by the container registry role assignment, run the following command:

Azure CLI

```
az role assignment list --scope  
/subscriptions/<subscriptionID>/resourceGroups/<resourcegroupname>/providers/  
Microsoft.ContainerRegistry/registries/<acrname> -o table
```

3. Compare the two service principals. If they don't match, integrate the AKS cluster with the container registry again.

Solution 4: Make sure the kubelet identity is referenced in the AKS VMSS

When a managed identity is used for authentication with the ACR, the managed identity is known as the kubelet identity. By default, the kubelet identity is assigned at the AKS VMSS level. If the kubelet identity is removed from the AKS VMSS, the AKS nodes can't pull images from the ACR.

To find the kubelet identity of your AKS cluster, run the following command:

Azure CLI

```
az aks show --resource-group <MyResourceGroup> --name <MyManagedCluster> --query  
identityProfile.kubeletIdentity
```

Then, you can list the identities of the AKS VMSS by opening the VMSS from the node resource group and selecting **Identity > User assigned** in the Azure portal or by running the following command:

Azure CLI

```
az vmss identity show --resource-group <NodeResourceGroup> --name <AksVmssName>
```

If the kubelet identity of your AKS cluster isn't assigned to the AKS VMSS, assign it back.

ⓘ Note

[Modifying the AKS VMSS using the IaaS APIs or from the Azure portal isn't supported](#), and no AKS operation can remove the kubelet identity from the AKS VMSS. This means that something unexpected removed it, for example, a manual removal performed by a team member. To prevent such removal or modification, you can consider using the [NRGLockdown feature](#).

Because modifications to the AKS VMSS aren't supported, they don't propagate at the AKS level. To reassign the kubelet identity to the AKS VMSS, a reconciliation operation is needed. To

do this, run the following command:

Azure CLI

```
az aks update --resource-group <MyResourceGroup> --name <MyManagedCluster>
```

Solution 5: Make sure the service principal is correct and the secret is valid

If you pull an image by using an [image pull secret](#), and that Kubernetes secret was created by using the values of a service principal, make sure that the associated service principal is correct and the secret is still valid. Follow these steps:

1. Run the following [kubectl get](#) and [base64](#) command to see the values of the Kubernetes secret:

Console

```
kubectl get secret <secret-name> --output="jsonpath=.data.\.dockerconfigjson" | base64 --decode
```

2. Check the expiration date by running the following [az ad sp credential list](#) command. The username is the service principal value.

Azure CLI

```
az ad sp credential list --id "<username>" --query "[].endDate" --output tsv
```

3. If necessary, reset the secret of that service principal by running the following [az ad sp credential reset](#) command:

Azure CLI

```
az ad sp credential reset --name "$SP_ID" --query password --output tsv
```

4. Update or re-create the Kubernetes secret accordingly.

Solution 6: Make sure the Kubernetes secret has the correct values of the container registry admin account

If you pull an image by using an [image pull secret](#), and that Kubernetes secret was created by using values of [container registry admin account](#), make sure that the values in the

Kubernetes secret are the same as the values of the container registry admin account. Follow these steps:

1. Run the following [kubectl get](#) and [base64](#) command to see the values of the Kubernetes secret:

Console

```
kubectl get secret <secret-name> --output="jsonpath=.data.\.dockerconfigjson" | base64 --decode
```

2. In the [Azure portal](#), search for and select **Container registries**.
3. In the list of container registries, select your container registry.
4. In the navigation pane for the container registry, select **Access keys**.
5. In the **Access keys** page for the container registry, compare the container registry values with the values in the Kubernetes secret.
6. If the values don't match, update or re-create the Kubernetes secret accordingly.

! Note

If a **Regenerate** password operation occurred, an operation that's named "Regenerate Container Registry Login Credentials" will be displayed in the **Activity log** page of the container registry. The **Activity log** has a [90-day retention period](#).

Cause 1b: 401 Unauthorized error due to incompatible architecture

You might encounter a "401 Unauthorized" error even when the AKS cluster identity is authorized (as described in the [Cause 1a: 401 Unauthorized error due to incorrect authorization](#) section). This issue can happen if the container image in the ACR doesn't match the architecture (such as arm64 versus amd64) of the node running the container. For example, deploying an arm64 image on an amd64 node or vice versa can result in this error.

The error message will appear as follows:

```
Failed to pull image "<acrname>.azurecr.io/<repository:\tag>": [rpc error: code = NotFound desc = failed to pull and unpack image "<acrname>.azurecr.io/<repository:\tag>": no match for platform in manifest: not found, failed to pull and unpack image "<acrname>.azurecr.io/<repository:\tag>": failed to resolve
```

```
reference "<acrname>.azurecr.io/<repository:tag>": failed to authorize: failed to fetch  
anonymous token: unexpected status from GET request to  
https://<acrname>.azurecr.io/oauth2/token?  
scope=repository%3A<repository>%3Apull&service=<acrname>.azurecr.io: 401  
Unauthorized]
```

When diagnosing this issue using the Azure CLI, you might see an unexpected "exec format error" if your system node pool runs a different architecture than the image in the ACR:

Azure CLI

```
az aks check-acr --resource-group <MyResourceGroup> --name <MyManagedCluster> --  
acr <myacr>.azurecr.io  
  
exec /canipull: exec format error
```

Solution: Push images with the correct architecture or push multi-architecture images

To resolve this issue, use one of the following methods:

- Ensure the container images pushed to ACR match the architecture of your AKS nodes (for example, arm64 or amd64).
- Create and push multi-architecture images that support both arm64 and amd64 architectures.

Cause 2: Image not found error

```
Failed to pull image "<acrname>.azurecr.io/<repository:tag>": [rpc error: code = NotFound  
desc = failed to pull and unpack image "<acrname>.azurecr.io/<repository:tag>": failed to  
resolve reference "<acrname>.azurecr.io/<repository:tag>":  
<acrname>.azurecr.io/<repository:tag>: not found
```

Solution: Make sure image name is correct

If you see this error, make sure that the image name is fully correct. You should check the registry name, registry login server, the repository name, and the tag. A common mistake is that the login server is specified as "azureacr.io" instead of "azurecr.io".

If the image name isn't fully correct, the [401 Unauthorized error](#) may also occur because AKS always tries anonymous pull regardless of whether the container registry has enabled

anonymous pull access.

Cause 3: 403 Forbidden error

Failed to pull image "<acrname>.azurecr.io/<repository:tag>": rpc error: code = Unknown desc = failed to pull and unpack image "<acrname>.azurecr.io/<repository:tag>": failed to resolve reference "<acrname>.azurecr.io/<repository:tag>": failed to authorize: failed to fetch anonymous token: **unexpected status: 403 Forbidden**

Solution 1: Make sure AKS virtual network link is set in the container registry's Private DNS zone

If the network interface of the container registry's private endpoint and the AKS cluster are in different virtual networks, make sure that the [virtual network link](#) for the AKS cluster's virtual network is set in the Private DNS zone of the container registry. (That link is named "privatelink.azurecr.io" by default.) If the virtual network link isn't in the Private DNS zone of the container registry, add it by using one of the following ways:

- In the Azure portal, select the private DNS zone "privatelink.azurecr.io", select **Virtual network links > Add** under the **Settings** panel, and then select a name and the virtual network of the AKS cluster. Select **OK**.

 **Note**

It's optional to select the "[Enable auto registration](#)" feature.

- [Create a virtual network link to the specified Private DNS zone by using Azure CLI.](#)

Solution 2: Add AKS Load Balancer's public IP address to allowed IP address range of the container registry

If the AKS cluster connects publicly to the container registry (NOT through a private link or an endpoint) and the public network access of the container registry is limited to selected networks, add AKS Load Balancer's public IP address to the allowed IP address range of the container registry:

1. Verify that the public network access is limited to selected networks.

In the Azure portal, navigate to the container registry. Under **Settings**, select **Networking**. On the **Public access** tab, **Public network access** is set to **Selected networks** or **Disabled**.

2. Obtain the AKS Load Balancer's public IP address by using one of the following ways:

- In the Azure portal, navigate to the AKS cluster. Under **Settings**, select **Properties**, select one of the virtual machine scale sets in the infrastructure resource group, and check the public IP address of the AKS Load Balancer.
- Run the following command:

```
Azure CLI
```

```
az network public-ip show --resource-group <infrastructure-resource-group> --name <public-IP-name> --query ipAddress -o tsv
```

3. Allow access from the AKS Load Balancer's public IP address by using one of the following ways:

- Run `az acr network-rule add` command as follows:

```
Azure CLI
```

```
az acr network-rule add --name acrname --ip-address <AKS-load-balancer-public-IP-address>
```

For more information, see [Add network rule to registry](#).

- In the Azure portal, navigate to the container registry. Under **Settings**, select **Networking**. On the **Public access** tab, under **Firewall**, add the AKS Load Balancer's public IP address to **Address range** and then select **Save**. For more information, see [Access from selected public network - portal](#).

 **Note**

If **Public network access** is set to **Disabled**, switch it to **Selected networks** first.

The screenshot shows the 'Networking' settings for an Azure Container Registry (ACR). Under 'Public access', the 'Selected networks' option is chosen. In the 'Firewall' section, there is a checkbox labeled 'Add your client IP address' and an 'Address range' input field containing 'Add AKS Load Balancer's IP here'. The 'Networking' tab is highlighted in the left sidebar.

Cause 4: "i/o timeout error"

Failed to pull image "<acrname>.azurecr.io/<repository:tag>": rpc error: code = Unknown desc = failed to pull and unpack image "<acrname>.azurecr.io/<repository:tag>": failed to resolve reference "<acrname>.azurecr.io/<repository:tag>": failed to do request: Head "https://<acrname>.azurecr.io/v2/<repository>/manifests/v1": dial tcp <acrprivateipaddress>:443: i/o timeout

! Note

The "i/o timeout" error occurs only when you [connect privately to a container registry by using Azure Private Link](#).

Solution 1: Make sure virtual network peering is used

If the network interface of the container registry's private endpoint and the AKS cluster are in different virtual networks, make sure that [virtual network peering](#) is used for both virtual networks. You can check virtual network peering by running the Azure CLI command `az network vnet peering list --resource-group <MyResourceGroup> --vnet-name <MyVirtualNetwork> --output table` or in the Azure portal by selecting the **VNETs > Peering** under the **Settings** panel. For more information about listing all peerings of a specified virtual network, see [az network vnet peering list](#).

If the virtual network peering is used for both virtual networks, make sure that the status is "Connected". If the status is [Disconnected](#), delete the peering from both virtual networks, and

then re-create it. If the status is "Connected", see the troubleshooting guide: [The peering status is "Connected"](#).

For further troubleshooting, connect to one of the AKS nodes or [pods](#), and then test the connectivity with the container registry at TCP level by using the Telnet or Netcat utility. Check the IP address with the `nslookup <acrname>.azurecr.io` command, and then run the `telnet <ip-address-of-the-container-registry> 443` command.

For more information about connecting to AKS nodes, see [Connect with SSH to Azure Kubernetes Service \(AKS\) cluster nodes for maintenance or troubleshooting](#).

Solution 2: Use Azure Firewall Service

If the network interface of the container registry's private endpoint and the AKS cluster are in different virtual networks, in addition to virtual network peering, you may use Azure Firewall Service to set up a [Hub-spoke network topology in Azure](#). When you set up the firewall rule, you need to use network rules to explicitly allow the [outbound connection](#) to the container registry private endpoint IP addresses.

Cause 5: No match for platform in manifest

The host operating system (node OS) is incompatible with the image that's used for the pod or container. For example, if you schedule a pod to run a Linux container on a Windows node, or a Windows container on a Linux node, the following error occurs:

```
Failed to pull image "<acrname>.azurecr.io/<repository:tag>":  
[  
  rpc error:  
  code = NotFound  
  desc = failed to pull and unpack image "<acrname>.azurecr.io/<repository:tag>": no  
  match for platform in manifest: not found,  
]
```

This error can occur for an image that's pulled from any source, as long as the image is incompatible with the host OS. The error isn't limited to images that are pulled from the container registry.

Solution: Configure the nodeSelector field correctly in your pod or deployment

Specify the correct `nodeSelector` field in the configuration settings of your pod or deployment. The correct value for this field's `kubernetes.io/os` setting ensures that the pod will be scheduled on the correct type of node. The following table shows how to set the `kubernetes.io/os` setting in YAML:

[] [Expand table](#)

Container type	YAML setting
Linux container	<code>"kubernetes.io/os": linux</code>
Windows container	<code>"kubernetes.io/os": windows</code>

For example, the following YAML code describes a pod that needs to be scheduled on a Linux node:

YAML
<pre>apiVersion: v1 kind: Pod metadata: name: aspnetapp labels: app: aspnetapp spec: containers: - image: "mcr.microsoft.com/dotnet/core/samples:aspnetapp" name: aspnetapp-image ports: - containerPort: 80 protocol: TCP nodeSelector: "kubernetes.io/os": linux</pre>

Cause 6: Kubelet exceeds the default image pull rate limit

When multiple jobs pull the same images, the Kubelet default pull rate limit might be exceeded. In this case, an error message like the following one is displayed:

Failed to pull image "acrname.azurecr.io/repo/nginx:latest": **pull QPS exceeded**. This occurred for pod <podname> at 4/22/2025, 12:48:32.000 PM UTC.

For more information about the limit, see the [registryPullQPS configuration](#).

Solution: Change the value of imagePullPolicy to IfNotPresent

To resolve this issue, change the value of [imagePullPolicy](#) from `Always` to `IfNotPresent` in the deployment YAML file to prevent unnecessary pulls. `IfNotPresent` ensures that the image is pulled from the registry only if it's not already present on the node.

Here's an example of the deployment YAML file:

YAML

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
        - name: my-container
          image: myacr.azurecr.io/my-image:latest
          imagePullPolicy: IfNotPresent
```

More information

If the troubleshooting guidance in this article doesn't help you resolve the issue, here are some other things to consider:

- Check the network security groups and route tables associated with subnets, if you've got any of those items.
- If a virtual appliance like a firewall controls the traffic between subnets, check the firewall and [Firewall access rules](#).

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Troubleshoot Azure App Configuration extension installation errors

Article • 10/15/2024

This article discusses some common error scenarios that you might encounter when you install or update the [Azure App Configuration extension](#) for Microsoft Azure Kubernetes Service (AKS).

ⓘ Note

If the Azure App Configuration extension was successfully installed but you experience issues while using it, see the [Azure App Configuration Kubernetes Provider troubleshooting guide](#).

Scenario 1: Azure App Configuration Kubernetes Provider is already installed

You try to install the Azure App Configuration extension for AKS, but you receive an error message that indicates that the Azure App Configuration Kubernetes Provider is already installed through the `helm install` command. The error message might resemble either of the following error messages.

Message 1

```
(ExtensionOperationFailed) The extension operation failed with the following error:  
Error: [ InnerError: [Helm installation failed : Resource already existing in your  
cluster : Recommendation Manually delete the resource(s) that currently exist in  
your cluster and try installation again. To delete these resources run the following  
commands: kubectl delete {resource type} -n {resource namespace} {resource  
name} : InnerError [rendered manifests contain a resource that already exists. Unable  
to continue with install: CustomResourceDefinition  
"azureappconfigurationproviders.azureconfig.io" in namespace "" exists and cannot be  
imported into the current release: invalid ownership metadata; annotation validation  
error: key "meta.helm.sh/release-name" must equal "azureappconfig": current value  
is "azureappconfiguration.kubernetesprovider"; annotation validation error: key  
"meta.helm.sh/release-namespace" must equal "kube-system": current value is  
"azappconfig-system"]]] occurred while doing the operation : [Create] on the config,  
For general troubleshooting visit: https://aka.ms/k8s-extensions-TSG.
```

Message 2

(ExtensionOperationFailed) The extension operation failed with the following error:
Error: [InnerError: [Helm installation failed : Resource already existing in your cluster : Recommendation Manually delete the resource(s) that currently exist in your cluster and try installation again. To delete these resources run the following commands: `kubectl delete {resource type} -n {resource namespace} {resource name}` : InnerError [rendered manifests contain a resource that already exists. Unable to continue with install: ServiceAccount "az-appconfig-k8s-provider" in namespace "azappconfig-system" exists and cannot be imported into the current release: invalid ownership metadata; annotation validation error: key "meta.helm.sh/release-name" must equal "azureappconfig": current value is "azureappconfiguration.kubernetesprovider"]]] occurred while doing the operation : [Create] on the config, For general troubleshooting visit: <https://aka.ms/k8s-extensions-TSG>.

Solution 1: Uninstall Azure App Configuration Kubernetes Provider first

Uninstall the Azure App Configuration Kubernetes Provider before you install the Azure App Configuration extension. For more information, see [Clean up resources](#).

Scenario 2: Targeted Azure App Configuration extension version doesn't exist

When you try to install the Azure App Configuration extension to [target a specific version](#), you receive an error message that states that the Azure App Configuration version doesn't exist:

(ExtensionOperationFailed) The extension operation failed with the following error:
Failed to resolve the extension version from the given values. Please refer <https://aka.ms/k8s-extension-type-versions> to find the correct version for your installation, For general troubleshooting visit: <https://aka.ms/k8s-extensions-TSG>.

Code: ExtensionOperationFailed

Message: The extension operation failed with the following error: **Failed to resolve the extension version from the given values.** Please refer <https://aka.ms/k8s-extension-type-versions> to find the correct version for your installation, For general troubleshooting visit: <https://aka.ms/k8s-extensions-TSG>.

Solution 2: Install again for a supported Azure App Configuration extension version

Try again to install the extension. Make sure that you use a [supported version of the Azure App Configuration extension](#).

Scenario 3: The targeted Azure App Configuration extension version exists but not in the specified region

Because Azure App Configuration extension aren't available in all AKS regions, you might receive the following error message:

(ExtensionTypeRegistrationGetFailed) Extension type microsoft.appconfiguration is not registered in region <region-name>.

Code: ExtensionTypeRegistrationGetFailed

Message: Extension type microsoft.appconfiguration is not registered in region <region-name>

Solution 3: Install in a different region

Run the installation in a [region](#) that the cluster extension supports.

Next steps

If you still experience installation issues, explore the [AKS troubleshooting guide](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Troubleshoot Azure Key Vault Secrets Provider add-on in AKS

Article • 12/13/2024

This article discusses how to troubleshoot issues that you might experience when using the [Azure Key Vault Secrets Provider add-on](#) in Azure Kubernetes Service (AKS).

ⓘ Note

This article applies to the AKS managed add-on version of the Azure Key Vault Secrets Provider. If you use the helm installed (self-managed) version, go to the [Azure Key Vault Provider for Secrets Store CSI Driver](#) GitHub documentation.

Prerequisites

- [Azure CLI](#)
- The Kubernetes [kubectl](#) tool (To install kubectl by using Azure CLI, run the [az aks install-cli](#) command.)
- The [Kubernetes Secrets Store CSI Driver](#) add-on, enabled on the AKS cluster
- The client URL ([curl](#)) tool
- The [Netcat](#) (`nc`) command-line tool for TCP connections

Troubleshooting checklist

Step 1: Confirm that Azure Key Vault Secrets Provider add-on is enabled on your cluster

Run the [az aks show](#) command to confirm that the add-on is enabled on your cluster:

Azure CLI

```
az aks show -g <aks-resource-group-name> -n <aks-name> --query  
'addonProfiles.azureKeyvaultSecretsProvider'
```

The command output should be similar to the following text:

Output

```
{  
  "config": null,  
  "enabled": true,  
  "identity": {  
    "clientId": "<client-id>",  
    "objectId": "<object-id>",  
    "resourceId": "/subscriptions/<subscription-  
id>/resourcegroups/<resource-group-  
name>/providers/Microsoft.ManagedIdentity/userAssignedIdentities/<azure-key-  
vault-secrets-provider-identity-name>"  
  }  
}
```

If the `enabled` flag is shown as `false` in the preceding output, the Azure Key Vault Secrets Provider add-on isn't enabled on your cluster. In this case, refer to [Azure Key Vault Provider for Secrets Store CSI Driver](#) GitHub documentation for further troubleshooting.

If the `enabled` flag is shown as `true` in the preceding output, the Azure Key Vault Secrets Provider add-on is enabled on your cluster. In this case, go to next steps in this article.

Step 2: Check the Secrets Store Provider and CSI Driver pod logs

Azure Key Vault Secrets Provider add-on logs are generated by both provider and driver pods. To troubleshoot issues that affect the provider or driver, examine the logs from the pod that's running on the same node as your application pod.

1. Run the [kubectl get](#) command to find the Secrets Store Provider and CSI Driver pods that run on the same node that your application pod runs on:

Bash

```
kubectl get pod -l 'app in (secrets-store-provider-azure, secrets-  
store-csi-driver)' -n kube-system -o wide
```

2. Run the [kubectl logs](#) command to view logs from the Secrets Store Provider pod:

Bash

```
kubectl logs -n kube-system <provider-pod-name> --since=1h | grep ^E
```

3. Run the [kubectl logs](#) command to view logs from the Secrets Store CSI Driver pod:

```
Bash
```

```
kubectl logs -n kube-system <csi-driver-pod-name> -c secrets-store --since=1h | grep ^E
```

Once you collect the Secrets Store Provider and CSI Driver pod logs, analyze these logs against the causes mentioned in the following sections to identify the issue and corresponding solution.

 **Note**

If you open a support request, it's a good idea to include the relevant logs from the Azure Key Vault Provider and the Secrets Store CSI Driver.

Cause 1: Couldn't retrieve the key vault token

You might see the following error entry in the logs or event messages:

```
Warning FailedMount 74s kubelet MountVolume.SetUp failed for volume "secrets-store-inline" : kubernetes.io/csi: mounter.SetupAt failed: rpc error: code = Unknown desc = failed to mount secrets store objects for pod default/test, err: rpc error: code = Unknown desc = failed to mount objects, error: failed to get keyvault client: failed to get key vault token: nmi response failed with status code: 404, err: <nil>
```

This error occurs because a Node Managed Identity (NMI) component in *aad-pod-identity* returned an error message about a token request.

Solution 1: Check the NMI pod logs

For more information about this error and how to resolve it, check the NMI pod logs, and refer to the [Microsoft Entra pod identity troubleshooting guide](#).

Cause 2: The provider pod can't access the key vault instance

You might see the following error entry in the logs or event messages:

```
E1029 17:37:42.461313 1 server.go:54] failed to process mount request, error:  
keyvault.BaseClient#GetSecret: Failure sending request: StatusCode=0 -- Original  
Error: context deadline exceeded
```

This error occurs because the provider pod can't access the key vault instance. Access might be prevented for any of the following reasons:

- A firewall rule is blocking egress traffic from the provider.
- Network policies that are configured in the AKS cluster are blocking egress traffic.
- The provider pods run on the host network. A failure might occur if a policy is blocking this traffic or if network jitters occur on the node.

Solution 2: Check network policies, allowlist, and node connection

To fix the issue, take the following actions:

- Put the provider pods on the allowlist.
- Check for policies that are configured to block traffic.
- Make sure that the node has connectivity to Microsoft Entra ID and your key vault.

To test the connectivity to your Azure key vault from the pod that's running on the host network, follow these steps:

1. Create the pod:

```
Bash

cat <<EOF | kubectl apply --filename -
apiVersion: v1
kind: Pod
metadata:
  name: curl
spec:
  hostNetwork: true
  containers:
    - args:
        - tail
        - -f
        - /dev/null
      image: curlimages/curl:7.75.0
      name: curl
  dnsPolicy: ClusterFirst
```

```
restartPolicy: Always  
EOF
```

- Run [kubectl exec](#) to run a command in the pod that you created:

```
Bash
```

```
kubectl exec --stdin --tty curl -- sh
```

- Authenticate by using your Azure key vault:

```
Bash
```

```
curl -X POST 'https://login.microsoftonline.com/<aad-tenant-id>/oauth2/v2.0/token' \  
-d 'grant_type=client_credentials&client_id=<azure-client-id>&client_secret=<azure-client-secret>&scope=https://vault.azure.net/.default'
```

- Try to get a secret that's already created in your Azure key vault:

```
Bash
```

```
curl -X GET 'https://<key-vault-name>.vault.azure.net/secrets/<secret-name>?api-version=7.2' \  
-H "Authorization: Bearer <access-token-acquired-above>"
```

Cause 3: The user-assigned managed identity is incorrect in the SecretProviderClass custom resource

If you encounter an HTTP error code "400" instance that's accompanied by an "Identity not found" error description, the user-assigned managed identity is incorrect in your `SecretProviderClass` custom resource. The full response resembles the following text:

```
Output
```

```
MountVolume.SetUp failed for volume "<volume-name>" :  
rpc error:  
code = Unknown desc = failed to mount secrets store objects for pod  
<namespace>/<pod>,  
err: rpc error: code = Unknown desc = failed to mount objects,  
error: failed to get objectType:secret, objectName:<key-vault-secret-name>, objectVersion:: azure.BearerAuthorizer#WithAuthorization:
```

```
Failed to refresh the Token for request to https://<key-vault-name>.vault.azure.net/secrets/<key-vault-secret-name>/?api-version=2016-10-01:  
    StatusCode=400 -- Original Error: adal: Refresh request failed.  
    Status Code = '400'.  
    Response body:  
{"error":"invalid_request","error_description":"Identity not found"}  
    Endpoint http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&client_id=<userAssignedIdentityID>&resource=https%!!(MISSING)A(MISSING)%!!(MISSING)F(MISSING)%!!(MISSING)F(MISSING)vault.azure.net
```

Solution 3: Update SecretProviderClass by using the correct userAssignedIdentityID value

Find the correct user-assigned managed identity, and then update the `SecretProviderClass` custom resource to specify the correct value in the `userAssignedIdentityID` parameter. To find the correct user-assigned managed identity, run the following `az aks show` command in Azure CLI:

Azure CLI

```
az aks show --resource-group <resource-group-name> \  
  --name <cluster-name> \  
  --query addonProfiles.azureKeyvaultSecretsProvider.identity.clientId \  
  --output tsv
```

For information about how to set up a `SecretProviderClass` custom resource in YAML format, see the [Use a user-assigned managed identity](#) section of the *Provide an identity to access the Azure Key Vault Provider for Secrets Store CSI Driver* article.

Cause 4: The Key Vault private endpoint is on a different virtual network than the AKS nodes

Public network access isn't allowed at the Azure Key Vault level, and the connectivity between AKS and Key Vault is made through a private link. However, the AKS nodes and the private endpoint of the Key Vault are on different virtual networks. This scenario generates a message that resembles the following text:

Output

```
MountVolume.SetUp failed for volume "<volume>" :  
  rpc error:  
    code = Unknown desc = failed to mount secrets store objects for pod
```

```
<namespace>/<pod>,
    err: rpc error: code = Unknown desc = failed to mount objects,
    error: failed to get objectType:secret, objectName: :<key-vault-secret-
name>, objectVersion:: keyvault.BaseClient#GetSecret:
        Failure responding to request:
            StatusCode=403 -- Original Error: autorest/azure: Service returned
an error.
            Status=403 Code="Forbidden"
            Message="Public network access is disabled and request is not from a
trusted service nor via an approved private link.\r\n
            Caller: appid=<application-id>;oid=<object-
id>;iss=https://sts.windows.net/<id>/;xms_mirid=/subscriptions/<subscription
-id>/resourcegroups/<aks-infrastructure-resource-
group>/providers/Microsoft.Compute/virtualMachineScaleSets/aks-<nodepool-
name>-<nodepool-id>-vmss;xms_az_rid=/subscriptions/<subscription-
id>/resourcegroups/<aks-infrastructure-resource-
group>/providers/Microsoft.Compute/virtualMachineScaleSets/aks-<nodepool-
name>-<nodepool-id>-vmss \r\n
            Vault: <keyvaultname>;location=<location>" InnerError=
{"code": "ForbiddenByConnection"}
```

Solution 4a: Set up a virtual network link and virtual network peering to connect the virtual networks

Fixing the connectivity issue is generally a two-step process:

- Create a [virtual network link](#) for the virtual network of the AKS cluster at the [private Azure DNS zone](#) level.
- Add [virtual network peering](#) between the virtual network of the AKS cluster and the virtual network of the Key Vault private endpoint.

These steps are described in more detail in the following sections.

Step 1: Create the virtual network link

Connect to the AKS cluster nodes to determine whether the fully qualified domain name (FQDN) of the Key Vault is resolved through a public IP address or a private IP address. If you receive the "Public network access is disabled and request is not from a trusted service nor via an approved private link" error message, the Key Vault endpoint is probably resolved through a public IP address. To check for this scenario, run the [nslookup](#) command:

Bash

```
nslookup <key-vault-name>.vault.azure.net
```

If the FQDN is resolved through a public IP address, the command output resembles the following text:

```
Console

root@aks-<nodepool-name>-<nodepool-id>-vmss<scale-set-instance>:/# nslookup
<key-vault-name>.vault.azure.net
Server:          168.63.129.16
Address:         168.63.129.16#53

Non-authoritative answer:
<key-vault-name>.vault.azure.net canonical name = <key-vault-
name>.privatelink.vaultcore.azure.net.
<key-vault-name>.privatelink.vaultcore.azure.net canonical name = data-
prod.weu.vaultcore.azure.net.
data-prod-weu.vaultcore.azure.net canonical name = data-prod-weu-
region.vaultcore.azure.net.
data-prod-weu-region.vaultcore.azure.net canonical name = azkms-prod-weu-
b.westeurope.cloudapp.azure.com.
Name:    azkms-prod-weu-b.westeurope.cloudapp.azure.com
Address: 20.1.2.3
```

In this case, create a virtual network link for the virtual network of the AKS cluster at the private DNS zone level. (A virtual network link is already created automatically for the virtual network of the Key Vault private endpoint.)

To create the virtual network link, follow these steps:

1. In the [Azure portal](#), search for and select **Private DNS zones**.
2. In the list of private DNS zones, select the name of your private DNS zone. In this example, the private DNS zone is **privatelink.vaultcore.azure.net**.
3. In the navigation pane of the private DNS zone, locate the **Settings** heading, and then select **Virtual network links**.
4. In the list of virtual network links, select **Add**.
5. In the **Add virtual network link** page, complete the following fields.

[] [Expand table](#)

Field name	Action
Link name	Enter a name to use for the virtual network link.
Subscription	Select the name of the subscription that you want to contain the virtual network link.

Field name	Action
Virtual network	Select the name of the virtual network of the AKS cluster.

6. Select the **OK** button.

After you finish the link creation procedure, run the `nslookup` command. The output should now resemble the following text that shows a more direct DNS resolution:

```
Console

root@aks-<nodepool-name>-<nodepool-id>-vmss<scale-set-instance>:/# nslookup
<key-vault-name>.vault.azure.net
Server:      168.63.129.16
Address:     168.63.129.16#53

Non-authoritative answer:
<key-vault-name>.vault.azure.net canonical name = <key-vault-
name>.privatelink.vaultcore.azure.net.
Name:   <key-vault-name>.privatelink.vaultcore.azure.net
Address: 172.20.0.4
```

After the virtual network link is added, the FQDN should be resolvable through a private IP address.

Step 2: Add virtual network peering between virtual networks

If you're using a private endpoint, you've probably disabled public access at the Key Vault level. Therefore, no connectivity exists between AKS and the Key Vault. You can test that configuration by using the following Netcat (nc) command:

```
Bash

nc -v -w 2 <key-vault-name>.vault.azure.net 443
```

If connectivity isn't available between AKS and the Key Vault, you see output that resembles the following text:

```
Output

nc: connect to <key-vault-name>.vault.azure.net port 443 (tcp) timed out:
Operation now in progress
```

To establish connectivity between AKS and the Key Vault, add virtual network peering between the virtual networks by following these steps:

1. Go to the [Azure portal](#).
2. Use one of the following options to follow the instructions from the [Create virtual network peer](#) section of the *Tutorial: Connect virtual networks with virtual network peering using the Azure portal* article to peer the virtual networks and verify that the virtual networks are connected (from one end):
 - Go to your AKS virtual network, and peer it to the virtual network of the Key Vault private endpoint.
 - Go to the virtual network of the Key Vault private endpoint, and peer it to the AKS virtual network.
3. In the Azure portal, search for and select the name of the other virtual network (the virtual network that you peered to in the previous step).
4. In the virtual network navigation pane, locate the **Settings** heading, and then select **Peerings**.
5. In the virtual network peering page, verify that the **Name** column contains the **Peering link name** of the **Remote virtual network** that you specified in step 2. Also, make sure that the **Peering status** column for that peering link has a value of **Connected**.

After you complete this procedure, you can run the Netcat command again. The DNS resolution and connectivity between AKS and the Key Vault should now succeed. Also, make sure that the Key Vault secrets are successfully mounted and work as expected, as shown by the following output:

```
Output

Connection to <key-vault-name>.vault.azure.net 443 port [tcp/https]
succeeded!
```

Solution 4b: Troubleshoot error code 403

Troubleshoot error code "403" by reviewing the [HTTP 403: Insufficient Permissions](#) section of the [Azure Key Vault REST API Error Codes](#) reference article.

Cause 5: The secrets-store.csi.k8s.io driver is missing from the list of registered CSI drivers

If you receive the following error message about a missing `secrets-store.csi.k8s.io` driver in the pod events, then the Secrets Store CSI Driver pods aren't running on the node in which the application is running:

```
Warning FailedMount 42s (x12 over 8m56s) kubelet, akswin000000
MountVolume.SetUp failed for volume "secrets-store01-inline" : kubernetes.io/csi:
mounter.SetUpAt failed to get CSI client: driver name secrets-store.csi.k8s.io not
found in the list of registered CSI drivers
```

Solution 5: Troubleshoot the Secret Store CSI Driver pod running on the same node

Retrieve the status of the Secret Store CSI Driver pod running on the same node by running the following command:

```
Bash
```

```
kubectl get pod -l app=secrets-store-csi-driver -n kube-system -o wide
```

If pod status isn't `Running` or any of the containers in this pod isn't in `Ready` state, then proceed to check the logs for this pod by following the steps in [Check the Secrets Store Provider and CSI Driver pod logs](#).

Cause 6: SecretProviderClass not found

You might see the following event when describing your application pod:

```
Output
```

Events:

Type	Reason	Age	From	Message
---	---	---	---	-----
Warning	FailedMount	2s (x5 over 10s)	kubelet	
				MountVolume.SetUp failed for volume "xxxxxxx" : rpc error: code = Unknown
				desc = failed to get secretproviderclass xxxxxxxx/xxxxxx, error:
				SecretProviderClass.secrets-store.csi.x-k8s.io "xxxxxxxxxxxx" not found

This event indicates that the `SecretProviderClass` referenced in your pod's volume specification doesn't exist in the same namespace as your application pod.

Solution 6a: Create the missing SecretProviderClass resource

Make sure that the `SecretProviderClass` resource referenced in your pod's volume specification exists in the same namespace where your application pod is running.

Solution 6b: Modify your application pod's volume specification to reference the correct SecretProviderClass resource name

Edit your application pod's volume specification to reference the correct `SecretProviderClass` resource name:

```
Output

...
spec:
  containers:
  ...
  volumes:
    - name: my-volume
      csi:
        driver: secrets-store.csi.k8s.io
        readOnly: true
        volumeAttributes:
          secretProviderClass: "xxxxxxxxxx"
```

Cause 7: The request is unauthenticated

The request is unauthenticated for Key Vault, as indicated by a "401" error code.

Solution 7: Troubleshoot error code 401

Troubleshoot error code "401" by reviewing the "[HTTP 401: Unauthenticated Request](#)" section of the [Azure Key Vault REST API Error Codes](#) reference article.

Cause 8: The number of requests exceeds the stated maximum

The number of requests exceeds the stated maximum for the timeframe, as indicated by a "429" error code.

Solution 8: Troubleshoot error code 429

Troubleshoot error code "429" by reviewing the "[HTTP 429: Too Many Requests](#)" section of the [Azure Key Vault REST API Error Codes](#) reference article.

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?



Yes



No

[Provide product feedback](#)

Troubleshoot errors when deploying AKS cluster extensions

Article • 04/10/2024

This article discusses how to troubleshoot errors that occur when you deploy cluster extensions for Microsoft Azure Kubernetes Service (AKS).

Extension creation errors

Error: Unable to get a response from the agent in time

This error occurs if Azure services don't receive a response from the cluster extension agent. This situation might occur because the AKS cluster can't establish a connection with Azure.

Cause 1: The cluster extension agent and manager pods aren't initialized

The cluster extension agent and manager are crucial system components that are responsible for managing the lifecycle of Kubernetes applications. The initialization of the cluster extension agent and manager pods might fail because of the following problems:

- Resource limitations
- Policy restrictions
- Node taints, such as `NoSchedule`

Solution 1: Make sure that the cluster extension agent and manager pods work correctly

To resolve this issue, make sure that the cluster extension agent and manager pods are correctly scheduled and can start. If the pods are stuck in an unready state, check the pod description by running the following `kubectl describe pod` command to get more details about the underlying problems (for example, taints that prevent scheduling, insufficient memory, or policy restrictions):

Console

```
kubectl describe pod -n kube-system extension-operator-{id}
```

Here's a command output sample:

Output

kube-system	extension-agent-55d4f4795f-sqx7q	2/2
Running 0	2d19h	
kube-system	extension-operator-56c8d5f96c-nvt7x	2/2
Running 0	2d19h	

For ARC-connected clusters, run the following command to check the pod description:

Console

```
kubectl describe pod -n azure-arc extension-manager-{id}
```

Here's a command output sample:

Output

NAMESPACE	NAME	READY
STATUS	RESTARTS	AGE
azure-arc	cluster-metadata-operator-744f8bfbd4-7pssr	0/2
ImagePullBackOff	0 6d19h	
azure-arc	clusterconnect-agent-7557d99d5c-rtgqh	0/3
ImagePullBackOff	0 6d19h	
azure-arc	clusteridentityoperator-9b8b88f97-nr8hf	0/2
ImagePullBackOff	0 6d19h	
azure-arc	config-agent-6d5fd59b8b-khw2d	0/2
ImagePullBackOff	0 6d19h	
azure-arc	controller-manager-5bc97f7db6-rt2zs	0/2
ImagePullBackOff	0 6d19h	
azure-arc	extension-events-collector-7596688867-sqzv2	0/2
ImagePullBackOff	0 6d19h	
azure-arc	extension-manager-86bbb949-6s59q	0/3
ImagePullBackOff	0 6d19h	
azure-arc	flux-logs-agent-5f55888db9-wnr4c	0/1
ImagePullBackOff	0 6d19h	
azure-arc	kube-aad-proxy-646c475dcc-92b86	0/2
ImagePullBackOff	0 6d19h	
azure-arc	logcollector-5cbc659bfb-9v96d	0/1
ImagePullBackOff	0 6d19h	
azure-arc	metrics-agent-5794866b46-j9949	0/2
ImagePullBackOff	0 6d19h	
azure-arc	resource-sync-agent-6cf4cf7486-flgwc	0/2
ImagePullBackOff	0 6d19h	

When the cluster extension agent and manager pods are operational and healthy, they establish communication with Azure services to install and manage Kubernetes applications.

Cause 2: An issue affects the egress block or firewall

If the cluster extension agent and manager pods are healthy, and you still encounter the "Unable to get a response from the agent in time" error, an egress block or firewall issue probably exists. This issue might block the cluster extension agent and manager pods from communicating with Azure.

Solution 2: Make sure that networking prerequisites are met

To resolve this problem, make sure that you follow the networking prerequisites that are outlined in [Outbound network and FQDN rules for Azure Kubernetes Service \(AKS\) clusters](#).

Cause 3: The traffic is not authorized

The extension agent unsuccessfully tries calling to

`<region>.dp.kubernetesconfiguration.azure.com` data plane service endpoints. This failure generates an "Errorcode: 403, Message This traffic is not authorized" entry in the extension-agent pod logs.

```
Console

kubectl logs -n kube-system extension-agent-<pod-guid>
{
  "Message": "2024/02/07 06:04:43 \\"ErrorCode: 403, Message This traffic is not authorized., Target /subscriptions/<subscription-id>/resourceGroups/<resource-group-name>/provider/managedclusters/clusters/<cluster-name>/configurations/getPendingConfigs\\\"", "LogType": "ConfigAgentTrace", "LogLevel": "Information", "Environment": "prod", "Role": "ClusterConfigAgent", "Location": "<region>", "ArmId": "/subscriptions/<subscription-id>/resourceGroups/<resource-group-name>/providers/Microsoft.ContainerService/managedclusters/<cluster-name>", "CorrelationId": "", "AgentName": "ConfigAgent", "AgentVersion": "1.14.5", "AgentTimestamp": "2024/02/07 06:04:43.672" }
{
  "Message": "2024/02/07 06:04:43 Failed to GET configurations with err : {\u003cnil\u003e}", "LogType": "ConfigAgentTrace", "LogLevel": "Information", "Environment": "prod", "Role": "ClusterConfigAgent", "Location": "<region>", "ArmId": "/subscriptions/<subscription-id>/resourceGroups/<resource-group-name>/providers/Microsoft.ContainerService/managedclusters/<cluster-name>",
```

```
"CorrelationId": "", "AgentName": "ConfigAgent", "AgentVersion": "1.14.5",
"AgentTimestamp": "2024/02/07 06:04:43.672" }
```

This error occurs if a preexisting PrivateLinkScope exists in an extension's data plane for Azure Arc-enabled Kubernetes, and the virtual network (or private DNS server) is shared between Azure Arc-enabled Kubernetes and the AKS-managed cluster. This networking configuration causes AKS outbound traffic from the extension data plane to also route through the same private IP address instead of through a public IP address.

Run the following `nslookup` command in your AKS cluster to retrieve the specific private IP address that the data plane endpoint is resolving to:

Console

```
PS D:\> kubectl exec -it -n kube-system extension-agent-<pod-guid> --
nslookup <region>.dp.kubernetesconfiguration.azure.com
Non-authoritative answer:
<region>.dp.kubernetesconfiguration.azure.com      canonical name =
<region>.privatelink.dp.kubernetesconfiguration.azure.com
Name:   <region>.privatelink.dp.kubernetesconfiguration.azure.com
Address: 10.224.1.184
```

When you search for the private IP address in the Azure portal, the search results point to the exact resource: virtual network, private DNS zone, private DNS server, and so on. This resource has a private endpoint that's configured for the extension data plane for Azure Arc-enabled Kubernetes.

Solution 3.1: (Recommended) Create separate virtual networks

To resolve this problem, we recommend that you create separate virtual networks for Azure Arc-enabled Kubernetes and AKS computes.

Solution 3.2: Create a CoreDNS override

If the recommended solution isn't possible in your situation, create a CoreDNS override for the extension data plane endpoint to go over the public network. For more information about how to customize CoreDNS, see the "["Hosts plugin"](#) section of "Customize CoreDNS with Azure Kubernetes Service."

To create a CoreDNS override, follow these steps:

1. Find the public IP address of the extension data plane endpoint by running the `nslookup` command. Make sure that you change the region (for example, `eastus2euap`) based on the location of your AKS cluster:

Console

```
nslookup <region>.dp.kubernetesconfiguration.azure.com
Non-authoritative answer:
Name:   clusterconfig<region>.<region>.cloudapp.azure.com
Address: 20.39.12.229
Aliases: <region>.dp.kubernetesconfiguration.azure.com
          <region>.privatelink.dp.kubernetesconfiguration.azure.com
          <region>.dp.kubernetesconfiguration.trafficmanager.net
```

2. Create a backup of the existing coreDNS configuration:

Bash

```
kubectl get configmap -n kube-system coredns-custom -o yaml >
coredns.backup.yaml
```

3. Override the mapping for the regional (for example, `eastus2euap`) data plane endpoint to the public IP address. To do this, create a YAML file that's named `corednsms.yaml`, and then copy the following example configuration into the new file. (Make sure that you update the address and the host name by using the values for your environment.)

YAML

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: coredns-custom      # This is the name of the configuration map
  that you can overwrite with your changes.
  namespace: kube-system
data:
  extensionsdp.override: | # You can select any name here, but it must
  have the .override file name extension.
    hosts {
      20.39.12.229 <region>.dp.kubernetesconfiguration.azure.com
      fallthrough
    }
```

4. To create the ConfigMap, run the `kubectl apply` command, specifying the name of your YAML manifest file:

Bash

```
kubectl apply -f corednsms.yaml
```

5. To reload the ConfigMap and enable Kubernetes Scheduler to restart CoreDNS without downtime, run the [kubectl rollout restart](#) command:

```
Bash
```

```
kubectl -n kube-system rollout restart deployment coredns
```

6. Run the `nslookup` command again to make sure that the data plane endpoint resolves to the provided public IP address:

```
Console
```

```
kubectl exec -it -n kube-system extension-agent-55d4f4795f-nld9q --  
nslookup [region].dp.kubernetesconfiguration.azure.com  
Name: <region>.dp.kubernetesconfiguration.azure.com  
Address: 20.39.12.229
```

The extension agent pod logs should no longer log "Errorcode: 403, Message This traffic is not authorized" error entries. Instead, the logs should contain "200" response codes.

```
Console
```

```
kubectl logs -n kube-system extension-agent-{id}  
{ "Message": "GET configurations returned response code {200}", "LogType":  
"ConfigAgentTrace", "LogLevel": "Information", "Environment": "prod",  
"Role": "ClusterConfigAgent", "Location": "<region>", "ArmId":  
"/subscriptions/<subscription-id>/resourceGroups/<resource-group-  
name>/providers/Microsoft.ContainerService/managedclusters/<cluster-name>",  
"CorrelationId": "", "AgentName": "ConfigAgent", "AgentVersion": "1.14.5"  
}
```

Error: Extension pods can't be scheduled if all the node pools in the cluster are "CriticalAddonsOnly" tainted

When this error occurs, the following entry is logged in the extension agent log:

```
Extension Pod error: 0/2 nodes are available: 2 node(s) had untolerated taint  
{CriticalAddonsOnly: true}. Preemption: 0/2 nodes are available: 2 Preemption is not  
helpful for scheduling.
```

Cause

This error occurs when you try to enable extensions (such as the Distributed Application Runtime (DAPR)) on an AKS cluster that has `CriticalAddonsOnly` tainted node pools. In

this situation, the extension pods aren't scheduled on any node because no toleration exists for these taints.

To view the error situation, examine the extension pods to verify that they're stuck in a pending state:

```
Console

kubectl get po -n {namespace-name} -l app.kubernetes.io/name={name}

NAME                      READY   STATUS    RESTARTS   AGE
{podname}                  0/2     Pending   0          2d6h
```

Describe the pods to see that they can't be scheduled because of an unsupportable taint:

```
Console

kubectl describe po -n {namespace-name} {podname}

Events:
Type      Reason           Age     From            Message
----      -----           ----   ----            -----
Warning   FailedScheduling 18s    default-scheduler 0/2 nodes are
available: 2 node(s) had untolerated taint {CriticalAddonsOnly: true}.
preemption: 0/2 nodes are available: 2 Preemption is not helpful for
scheduling.
```

ⓘ Note

- We recommend that you don't install extensions on `CriticalAddonsOnly` tainted node pools unless doing this is required for application workloads.
- We recommend that you don't use a `CriticalAddonsOnly` taint on single node pool clusters. If you use that taint in a cluster that has just one node pool, you can't schedule application pods in the cluster. Make sure that at least one node pool in the cluster doesn't have this taint. For more information about when the `CriticalAddonsOnly` annotation should be used, see [Manage system node pools in Azure Kubernetes Service \(AKS\)](#).

Solution 1: Add a node pool to the cluster

To resolve this problem, add one more node pool that doesn't have a `CriticalAddonsOnly` taint. This action causes the extension pods to be scheduled on the new node pool.

Solution 2: Remove the "CriticalAddonsOnly" taint

If it's possible and practical, you can remove the `CriticalAddonsOnly` taint in order to install the extension on the cluster.

Helm errors

You might encounter any of the following Helm-related errors:

- [Timed out waiting for resource readiness](#)
- [Unable to download the Helm chart from the repo URL](#)
- [Helm chart rendering failed with given values](#)
- [Resource already exists in your cluster](#)
- [Operation is already in progress for Helm](#)

Error: Timed out waiting for resource readiness

The installation of a Kubernetes application fails and displays the following error messages:

| job failed: BackoffLimitExceeded

| Timed out waiting for the resource to come to a ready/completed state.

Cause

This problem has the following common causes:

- Resource constraints: Inadequate memory or CPU resources within the cluster can prevent the successful initialization of pods, jobs, or other Kubernetes resources. Eventually, this situation causes the installation to time out. Policy constraints or node taints (such as `NoSchedule`) can also block resource initialization.
- Architecture mismatches: Trying to schedule a Linux-based application on a Windows-based node (or vice-versa) can cause failures in Kubernetes resource initialization.

- Incorrect configuration settings: Incorrect configuration settings can prevent pods from starting.

Solution

To resolve this problem, follow these steps:

1. Check resources: Make sure that your Kubernetes cluster has sufficient resources, and that pod scheduling is permitted on the nodes (you should consider taints). Verify that memory and CPU resources meet the requirements.
2. Inspect events: Check the events within the Kubernetes namespace to identify potential problems that might prevent pods, jobs, or other Kubernetes resources from reaching a ready state.
3. Check Helm charts and configurations: Many Kubernetes applications use Helm charts to deploy resources on the cluster. Some applications might require user input through configuration settings. Make sure that all provided configuration values are accurate and meet the installation requirements.

Error: Unable to download the Helm chart from the repo URL

This error is caused by connectivity problems that occur between the cluster and the firewall in addition to egress blocking problems. To resolve this problem, see [Outbound network and FQDN rules for Azure Kubernetes Service \(AKS\) clusters](#).

Error: Helm chart rendering failed with given values

This error occurs if Kubernetes applications rely on Helm charts to deploy resources within the Kubernetes cluster. These applications might require user input that's provided through configuration settings that are passed as Helm values during installation. If any of these crucial configuration settings are missing or incorrect, the Helm chart might not render.

To resolve this problem, check the extension or application documentation to determine whether you omitted any mandatory values or provided incorrect values during the application installation. These guidelines can help you to fix Helm chart rendering problems that are caused by missing or inaccurate configuration values.

Error: Resource already exists in your cluster

This error occurs if a conflict exists between the Kubernetes resources within your cluster and the Kubernetes resources that the application is trying to install. The error message usually specifies the name of the conflicting resource.

If the conflicting resource is essential and can't be replaced, you might not be able to install the application. If the resource isn't critical and can be removed, delete the conflicting resource, and then try the installation again.

Error: Operation is already in progress for Helm

This error occurs if there's an operation already in progress for a particular release. To resolve this problem, wait 10 minutes, and then retry the operation.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?



Troubleshoot Dapr extension installation errors

Article • 03/17/2025

This article discusses some common error messages that you may receive when you install or update the [Distributed Application Runtime \(Dapr\)](#) extension for Microsoft Azure Kubernetes Service (AKS) or Arc for Kubernetes.

Learn more about the level of support provided for the Dapr extension.

Scenario 1: Installation fails but doesn't show an error message

If the extension generates an error message when you create or update it, you can inspect where the creation failed by running the `az k8s-extension list` command:

Azure CLI

```
az k8s-extension list --resource-group <my-resource-group-name> \
--cluster-name <my-cluster-name> \
--cluster-type managedClusters
```

If a wrong key is used in the configuration settings, such as `global.ha=false` instead of `global.enabled=false`, the following JSON status is returned. The error message is captured in the `message` property.

JSON

```
"statuses": [
{
  "code": "InstallationFailed",
  "displayStatus": null,
  "level": null,
  "message": "Error: {failed to install chart from path [] for release [dapr-1]: err [template: dapr/charts/dapr_sidecar_injector/templates/dapr_sidecar_injector_poddisruptionbudget.yaml:1:17: executing \\"dapr/charts/dapr_sidecar_injector/templates/dapr_sidecar_injector_poddisruptionbudget.yaml\\" at <.Values.global.enabled>: can't evaluate field enabled in type interface {}]} occurred while doing the operation : {Installing the extension} on the config",
  "time": null
}
```

```
    }  
],
```

Here's another example of a JSON error message:

JSON

```
"statuses": [  
  {  
    "code": "InstallationFailed",  
    "displayStatus": null,  
    "level": null,  
    "message": "The extension operation failed with the following error:  
unable to add the configuration with configId {extension:microsoft-dapr} due  
to error: {error while adding the CRD configuration: error {failed to get  
the immutable configMap from the elevated namespace with err: configmaps  
'extension-immutable-values' not found }}. (Code:  
ExtensionOperationFailed)",  
    "time": null  
  }  
]
```

Solution 1: Restart the cluster, register the service provider, or delete and reinstall Dapr

To fix this issue, try the following methods:

- Restart your AKS or Arc for Kubernetes cluster.
- Register the [KubernetesConfiguration service provider](#).
- Force delete and [reinstall the Dapr extension](#).

Scenario 2: Targeted Dapr version doesn't exist

When you try to install the Dapr extension to [target a specific version](#), you receive an error message that states that the Dapr version doesn't exist:

(ExtensionOperationFailed) The extension operation failed with the following error:
Failed to resolve the extension version from the given values.

Code: ExtensionOperationFailed

Message: The extension operation failed with the following error: Failed to resolve the extension version from the given values.

Solution 2: Install again for a supported Dapr version

Try again to install the extension. Make sure that you use a [supported version of Dapr](#).

Scenario 3: The targeted Dapr version exists but not in the specified region

Because some versions of Dapr aren't available in all regions, you might receive the following error message:

(ExtensionTypeRegistrationGetFailed) Extension type microsoft.dapr is not registered in region <regionname>.

Code: ExtensionTypeRegistrationGetFailed

Message: Extension type microsoft.dapr is not registered in region <regionname>

Solution 3: Install in a different region

Install in a [region in which your Dapr version is supported](#).

Scenario 4: Dapr is already installed

You try to install the Dapr extension for AKS or Arc for Kubernetes, but you receive an error message that indicates that the `dapr-system` namespace already exists. This error message resembles the following text:

(ExtensionOperationFailed) The extension operation failed with the following error:
Error: {failed to install chart from path [] for release [dapr-ext]: err [rendered manifests contain a resource that already exists. Unable to continue with install: ServiceAccount "dapr-operator" in namespace "dapr-system" exists and cannot be imported into the current release: invalid ownership metadata; annotation validation error: key "meta.helm.sh/release-name" must equal "dapr-ext": current value is "dapr"]}} occurred while doing the operation : {Installing the extension} on the config

Solution 4: Uninstall Dapr OSS first

Uninstall the Dapr OSS before you install the Dapr extension. For more information, see [Migrate from Dapr OSS to the Dapr extension for AKS](#).

Scenario 5: The placement server pod is in a bad state

You encounter the following error:

```
0/4 nodes are available: 1 node(s) were unschedulable, 3 node(s) had volume node affinity conflict. preemption: 0/4 nodes are available: 4 Preemption is not helpful for scheduling.
```

This issue might happen when the placement server pod tries to use the persistent volume that's created in a different zone from the placement server pod itself.

Solution 5: Install Dapr in multiple availability zones or limit the placement service to a particular availability zone

To resolve this issue, use one of the following methods:

- Follow the recommended approach in [Install Dapr in multiple availability zones while in HA mode](#).
- Limit the placement service to a particular availability zone by creating a custom storage class and using it for the placement service, and then run the following command:

Azure CLI

```
az k8s-extension create --cluster-type managedClusters
--cluster-name <clustername>
--resource-group <resourcegroup>
--name <name>
--extension-type Microsoft.Dapr
--auto-upgrade-minor-version <minorversion>
--version <version>
--configuration-settings
"dapr_placement.volumeclaims.storageClassName=zone-restricted"
```

Here's an example of creating a custom storage class:

YAML

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
```

```
name: zone-restricted
provisioner: disk.csi.azure.com
reclaimPolicy: Delete
allowVolumeExpansion: true
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
- matchLabelExpressions:
  - key: topology.kubernetes.io/zone
    values:
    - centralus-1
parameters:
storageaccounttype: StandardSSD_LRS
```

Next steps

If you're still experiencing installation issues, [create a support request ↗](#) for Microsoft to investigate and resolve.

If you're experiencing Dapr runtime security risks and regressions while using the extension, open an issue with the [Dapr open source project ↗](#).

 Note

Learn more about [how Microsoft handles issues raised for the Dapr extension](#).

You could also start a discussion in the Dapr project Discord:

- [Dapr runtime ↗](#)
- [Dapr components ↗](#)

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#)

Troubleshoot the failed deployment of a Kubernetes application offer

Article • 04/22/2025

This article discusses how to troubleshoot a failed deployment of a Kubernetes application offer that was accepted on the Microsoft Azure Marketplace. When you initiate the purchase of a Kubernetes offer, Azure deploys an Azure Resource Manager template (ARM template) that tries to install the required resources to fulfill the offer. However, the ARM template deployment might fail for various reasons.

Troubleshooting checklist

Examine the deployment operation logs

To determine the cause of the deployment failure, you have to examine the [deployment operation logs](#). If you're still viewing the **Your deployment failed** page in the Azure portal, begin at step 5 of the following procedure. If, instead, you exited the Azure portal or navigated to another portal page, follow all these steps:

1. In the [Azure portal](#), search for and select **Resource groups**.
2. In the list of resource groups, select the name of the resource group in which you tried to deploy the Kubernetes application.
3. On the **Overview** page of your resource group, locate the **Essentials** section, and then select the hyperlinked text that appears next to the **Deployments** field. This text displays the success rate of your resource group's resource deployment history (for example, **4 failed, 30 succeeded**).
4. In the list of attempted deployments for your resource group, select the **Deployment name** value of the deployment that failed, based on the following corresponding fields:
 - **Last modified** (a time stamp)
 - **Duration**
 - **Status** (shows **Failed** instead of **Succeeded**)
5. In the **Deployment details** list on the deployment page, locate the **Resource** for which the **Status** field has a value of **Conflict**. Select the **Operation details** link for that resource.

The screenshot shows the Microsoft Azure portal's deployment overview page. A deployment named "contoso.arc_billingextension_offer-20221101172157" has failed. The status is listed as "Conflict". A red box highlights the "Status message" field, which contains JSON error details.

6. In the **Operation details** pane, locate the **Status** property (shows a value of **Conflict**), and examine the **Status message** box below the property.

This screenshot shows the "Operation details" pane from the previous step. It displays various tracking and service request IDs. The "Status" field is set to "Conflict". The "Status message" field is highlighted with a red box and contains the following JSON code:

```

1 {
2   "status": "Failed",
3   "error": {
4     "code": "ExtensionOperationFailed",
5     "message": "The extension operation failed with the following error: Failed to resolve the extension version from the given values."
6   }
7 }
  
```

The JSON code within the status message shows a `status` property of `Failed`. It also shows an `error` property that contains the child properties of `code` (an error code name, such as "ExtensionOperationFailed") and `message` (an error message description, such as "The extension operation failed with the following error: Failed to resolve the extension version from the given values."). The JSON code resembles the following text:

JSON

```
{
  "status": "Failed",
  "error": {
    "code": "ExtensionOperationFailed",
    "message": "The extension operation failed with the following error: Failed to resolve the extension version from the given values."
  }
}
```

The following sections discuss the cause and solution for some common failure scenarios.

Cause 1: The application didn't install on the selected AKS cluster

If the Kubernetes application didn't install on the selected Azure Kubernetes Service (AKS) cluster, you receive an error message that resembles the following text:

Request failed to https://management.azure.com/subscriptions/<subscription-guid>/resourceGroups/resourceGroup/providers/Microsoft.ContainerService/managedclusters/aks-cluster/extensionaddons/default?api-version=2021-03-01. Error code: Forbidden.
Reason: Forbidden.

JSON

```
{  
  "error": {  
    "code": "AuthorizationFailed",  
    "message": "The client '<client-guid>' with object id '<client-guid>' does  
    not have authorization to perform action  
    'Microsoft.ContainerService/managedclusters/extensionaddons/read' over scope  
    '/subscriptions/<subscription-  
    guid>/resourceGroups/resourceGroup/providers/Microsoft.ContainerService/managed-  
    clusters/aks-cluster/extensionaddons/default' or the scope is invalid. If  
    access was recently granted, please refresh your credentials."  
  }  
}
```

Solution 1a: Register the Microsoft.KubernetesConfiguration resource provider

Register the Microsoft.KubernetesConfiguration resource provider. In this case, the installation failed because the Microsoft.KubernetesConfiguration resource provider is required for you to deploy the Kubernetes application. For registration instructions, see the "[Register resource providers](#)" section in the *Deploy a container offer from Azure Marketplace* article.

Solution 1b: Maintain the health of the AKS cluster

In general, you should [check the health of the AKS cluster](#) to prevent other issues from occurring during the installation period. To make sure that the cluster is healthy, resolve issues that are identified on the cluster.

Solution 1c: Examine the Azure Monitor activity log

What if the cluster is healthy, but the installation still fails? In that case, examine the [Azure Monitor activity log](#) within the AKS cluster to find the cause of the failure at that stage of the installation.

Cause 2: The subscription has resource constraints

Because your Azure subscription has resource constraints, you experience a failure that produces an error message that's similar to the following text:

The 'unknown' payment instrument(s) is not supported for offer with OfferId: '<offer-name>', PlanId '<subscription-plan-name>'.

Solution 2: Make sure your subscription meets the necessary billing configuration

Verify the subscription's billing configuration to make sure that it meets the resource requirements of the Kubernetes application. For more information, see [Purchase validation checks](#).

Cause 3: The offer wasn't available in your region

You receive an error message that states that the offer can't be sold in a certain geographical region. The error message might resemble the following text:

The Offer: '<offer-name>' cannot be purchased by subscription: '<subscription-guid>' as it is not to be sold in market: '<two-letter-region-code>'.

Solution 3: Recheck whether and where the offer is still available

Verify that the offer is still available, and double-check the regions that the offer applies to.

Cause 4: An internal server error occurred

The Kubernetes application didn't install because an extension resource didn't install. This failure generates the following error message:

Extension failed to deploy with Internal server error

Solution 4: Delete and reinstall the extension

First, delete the extension resource that's a part of the offer purchase. Then, reinstall the extension.

Cause 5: The Helm chart didn't install

Errors in the Helm chart generate the following error message:

Failed to install chart from path [] for release

Solution 5: Recheck the entries that you made in the ARM template

Make sure that the values and selections that you entered on the Azure portal for the ARM template deployment are acceptable in the Kubernetes application.

Cause 6: You haven't accepted the legal terms on the subscription for this plan

Before the subscription can be used, you need to accept the legal terms of the image. Otherwise, you get the following error message:

You have not accepted the legal terms on this subscription: '<subscription-guid>' for this plan. Before the subscription can be used, you need to accept the legal terms of the image.

Solution 6: Accept the legal terms

Portal

You can deploy through the [Azure portal](#). The Azure portal provides a UI experience for reading and accepting the legal terms.

Next steps

[Troubleshoot errors when deploying AKS cluster extensions](#)

[Third-party information disclaimer](#)

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the

performance or reliability of these products.

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Troubleshoot network isolated Azure Kubernetes Service (AKS) cluster issues

Article • 04/16/2025

This article discusses how to troubleshoot issues in [network isolated Azure Kubernetes Service \(AKS\) clusters](#).

Prerequisites

- The Kubernetes [kubectl](#) tool. You can install kubectl by running the [Azure CLI](#) command `az aks install-cli`.

Network isolated cluster support

The network isolated cluster follows a similar support model to other [AKS add-ons](#). When using a network isolated cluster with Azure Container Registry (ACR), you have two options:

- Bring Your Own (BYO) ACR
- AKS-managed ACR

If you choose BYO ACR, you're responsible for configuring your ACR and its associated resources properly.

Issue 1: Cluster image pull fails due to network isolation

Network isolated clusters use ACR cache rules for image pulls. If an image pull fails due to network isolation, follow these steps:

- For BYO ACR:

Verify that the private ACR resources are configured, including the cache rule and private endpoints. For more information about how to configure them, see steps 3 and 4 under the [Deploy a network isolated cluster with bring your own ACR](#) section.
- For AKS-managed ACR:
 - By default, only Microsoft Container Registry (MCR) images are supported. If the image pull failure occurs with MCR images, check if the associated ACR and private endpoint resource named with the keyword `bootstrap` exist. If they don't exist, reconcile the cluster.

- If the image pull failure occurs with images from other registries, create extra cache rules in the private ACR for those images.

Issue 2: Cluster image pull fails after updating an existing cluster to a network isolated cluster or updating the private ACR resource ID

The failure is an intended behavior. To resolve this issue, reimagine the node to update the kubelet configuration in Container Service Extension (CSE) following the update actions in [Update your ACR ID](#).

Issue 3: ACR or associated cache rules, private endpoints, or private DNS zones are deleted

If the cache rule is deleted from the managed ACR accidentally, the mitigation is to delete the ACR and then reconcile the cluster. If the ACR itself, the associated private endpoints, or the associated private DNS zones are deleted accidentally, the mitigation is just to reconcile the cluster.

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Troubleshoot managed namespace errors

08/18/2025

This article provides guidance for resolving errors that occur in Microsoft Azure Kubernetes Service (AKS) when you use the `ManagedNamespacePreview` flag.

Prerequisites

The following tools must be installed and configured:

- [Azure CLI](#)
- [kubectl](#), the Kubernetes command-line client

Error 1: Feature not found

Symptom

When you try to register the `ManagedNamespacePreview` flag, you receive the following error message:

(FeatureNotFound) The feature '<feature_name>' could not be found.

Resolution

Make sure that the command is entered correctly and uses correct spelling.

To register this preview feature, run the `az feature register` command:

```
az feature register --namespace Microsoft.containerService -n ManagedNamespacePreview
```

Error 2 - Can't create a managed namespace

Symptom

When you try to create a managed namespace, you receive the following error message:

*(BadRequest) Managed namespace requires feature flag
Microsoft.ContainerService/ManagedNamespacePreview to be registered.*

Resolution

This message indicates that the feature flag is not yet registered. If you already ran the command to register the flag, verify the registration status by running the the `az feature show` command:

```
az feature show --namespace Microsoft.ContainerService -n ManagedNamespacePreview
```

Error 3: Can't use or change some namespaces

Symptom

You receive the following error message:

The namespace name cannot be the same as the name of a system namespace.

Resolution

Users are not allowed to change certain namespaces or create managed namespaces that use certain names because the names are used by system components or resources. This list includes the following namespaces:

- default
- kube-system
- kube-node-lease
- kube-public
- gatekeeper-system
- cert-manager
- calico-system
- tigera-system
- app-routing-system
- aks-istio-system
- istio-system
- dapr-system
- flux-system
- prometheus-system
- eraser-system

Error 4 - Can't update or delete managed namespaces

Symptom

You can't create, update, or delete managed namespaces.

Resolution

This behavior is by design. Users are not allowed to create, update, or delete managed namespaces if the managed cluster is not in a running state.

Error 5: Can't use some kubectl commands

Symptom

When you try to modify a managed namespace through kubectl, you receive the following error message:

*Updating resource quota defaultresourcequota is not allowed because it is managed by ARM.
Please update this resource quota though ARM api.*

Resolution

Because a managed namespace is managed by Microsoft Azure Resource Manager (ARM), changes to its metadata (labels and annotations) through kubectl commands are restricted. The affected actions include, but are not limited to, edits and deletions to:

- A managed namespace through `kubectl edit ns <namespace-name>` or `kubectl delete ns <namespace name>`
- A namespace through `kubectl delete resourcequota defaultresourcequota --namespace <namespace-name>`
- A defaultresourcequota through `kubectl delete resourcequota defaultresourcequota --namespace <namespace-name>`
- A defaultnetworkpolicy through `kubectl delete networkpolicy defaultnetworkpolicy --namespace <namespace-name>`

Modifications to namespaces must be made through the ARM API to maintain consistency with the managed state. You can manage your managed namespaces in the Azure portal or through [CLI commands](#).

[Third-party information disclaimer](#)

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the Kubernetes Event-driven Autoscaling add-on

Article • 08/13/2024

This article discusses how to troubleshoot the Kubernetes Event-driven Autoscaling (KEDA) add-on to the Microsoft Azure Kubernetes Service (AKS). When you deploy the KEDA AKS add-on, you might experience problems that are associated with the configuration of the application autoscaler. This article will help you troubleshoot errors and resolve common problems that affect the add-on but aren't covered in the official KEDA [FAQ](#) and [troubleshooting guide](#).

Prerequisites

- The Kubernetes [kubectl](#) tool. To install kubectl by using the [Azure CLI](#), run the `az aks install-cli` command.

KEDA add-on support

The KEDA add-on follows a similar support model to other [AKS add-ons](#). All [Azure KEDA scalers](#) are supported, but AKS doesn't support third-party scalers. If you encounter issues with third-party scalers, open an issue in the official [KEDA GitHub repository](#).

Troubleshooting checklist

Verify and troubleshoot KEDA components by using the instructions in the following sections.

Check the available KEDA version

You can determine the available KEDA version by using the [kubectl get](#) command:

Bash

```
kubectl get crd/scaledobjects.keda.sh -o custom-columns='APP:.metadata.labels.app\.kubernetes\.io/version'
```

The command output displays the installed version of KEDA:

Output

Make sure that the cluster firewall is configured correctly

KEDA might not scale applications successfully because it can't start.

When you check the operator logs, you might find error entries that resemble the following text:

Output

```
1.6545953013458195e+09 ERROR Failed to get API Group-Resources {"error": "Get \"https://10.0.0.1:443/api?timeout=32s\": EOF"}  
sigs.k8s.io/controller-runtime/pkg/cluster.New  
/go/pkg/mod/sigs.k8s.io/controller-  
runtime@v0.11.2/pkg/cluster/cluster.go:160  
sigs.k8s.io/controller-runtime/pkg/manager.New  
/go/pkg/mod/sigs.k8s.io/controller-  
runtime@v0.11.2/pkg/manager/manager.go:313  
main.main  
/workspace/main.go:87  
runtime.main  
/usr/local/go/src/runtime/proc.go:255  
1.6545953013459463e+09 ERROR setup unable to start manager {"error": "Get  
\"https://10.0.0.1:443/api?timeout=32s\": EOF"}  
main.main  
/workspace/main.go:97  
runtime.main  
/usr/local/go/src/runtime/proc.go:255
```

In the metric server section, you might discover that KEDA can't start:

Output

```
I0607 09:53:05.297924 1 main.go:147] keda_metrics_adapter "msg"="KEDA  
Version: 2.7.1"  
I0607 09:53:05.297979 1 main.go:148] keda_metrics_adapter "msg"="KEDA  
Commit: "  
I0607 09:53:05.297996 1 main.go:149] keda_metrics_adapter "msg"="Go Version:  
go1.17.9"  
I0607 09:53:05.298006 1 main.go:150] keda_metrics_adapter "msg"="Go OS/Arch:  
linux/amd64"  
E0607 09:53:15.344324 1 logr.go:279] keda_metrics_adapter "msg"="Failed to  
get API Group-Resources" "error"="Get \"https://10.0.0.1:443/api?  
timeout=32s\": EOF"  
E0607 09:53:15.344360 1 main.go:104] keda_metrics_adapter "msg"="failed to  
setup manager" "error"="Get \"https://10.0.0.1:443/api?timeout=32s\": EOF"  
E0607 09:53:15.344378 1 main.go:209] keda_metrics_adapter "msg"="making
```

```
provider" "error"="Get \"https://10.0.0.1:443/api?timeout=32s\": EOF"
E0607 09:53:15.344399 1 main.go:168] keda_metrics_adapter "msg"="unable to
run external metrics adapter" "error"="Get \"https://10.0.0.1:443/api?
timeout=32s\": EOF"
```

This scenario probably means that the KEDA add-on isn't able to start because of a misconfigured firewall. To make sure that KEDA runs correctly, configure the firewall to meet the [Azure Global required network rules](#).

Enable the add-on for clusters with self-managed open-source KEDA installations

In theory, you can install KEDA many times, even though Kubernetes allows only one metric server to be installed. However, we don't recommend multiple installations because only one installation would work.

When the KEDA add-on is installed on an AKS cluster, the previous installation of open-source KEDA will be overridden, and the add-on will take over. In this scenario, the customization and configuration of the self-installed KEDA deployment will be lost and no longer applied.

Though the existing autoscaling might continue to be operational, this situation introduces a risk. The KEDA add-on will be configured differently, and it won't support features such as managed identity. To prevent errors during the installation, we recommend that you uninstall existing KEDA installations before you enable the KEDA add-on.

To determine which metrics adapter is used by KEDA, run the `kubectl get` command:

Bash

```
kubectl get APIService/v1beta1.external.metrics.k8s.io -o custom-
columns='NAME:.spec.service.name,NAMESPACE:.spec.service.namespace'
```

An overview shows the service and namespace that Kubernetes will use to get metrics:

Output

NAME	NAMESPACE
keda-operator-metrics-apiserver	kube-system

 **Warning**

If the namespace isn't `kube-system`, the AKS add-on is being ignored, and another metric server is being used.

How to restart KEDA operator pods when workload identity isn't injected properly

If you're using [Microsoft Entra Workload ID](#) and you enable KEDA before the Workload ID is used, you must restart the KEDA operator pods. This ensures the correct environment variables are injected. To do this, follow these steps:

1. Restart the pods by running the following command:

```
Bash
```

```
kubectl rollout restart deployment keda-operator -n kube-system
```

2. Obtain KEDA operator pods by running the following command, and then locate pods with names starting with 'keda-operator'.

```
Bash
```

```
kubectl get pod -n kube-system
```

3. To verify that the environment variables have been successfully injected, run the following command:

```
Bash
```

```
kubectl describe pod <keda-operator-pod-name> -n kube-system
```

If the variables have been successfully injected, you should see values for `AZURE_TENANT_ID`, `AZURE_FEDERATED_TOKEN_FILE`, and `AZURE_AUTHORITY_HOST` in the **Environment** section.

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

Breaking changes in Event-driven Autoscaling add-ons 2.15 and 2.14

Article • 03/03/2025

This article explores the key breaking changes that are introduced in Event-driven Autoscaling (KEDA) add-ons 2.15 and 2.14. It also provides information about how to prepare your Azure Kubernetes Service (AKS) clusters for these updates.

Breaking changes in KEDA 2.15 and 2.14

Your AKS cluster Kubernetes version determines which KEDA version will be installed on your AKS cluster. For AKS clusters that are running Kubernetes version 1.30, KEDA add-on version 2.14 is installed. The upcoming release of Kubernetes 1.32 on Azure will ship together with KEDA add-on version 2.15.

KEDA 2.15

- Removal of pod-managed identities support [↗](#).

If you're using pod-managed identities, we recommend that you migrate to [workload identity for authentication](#).

KEDA 2.14

- Removal of Azure Data Explorer 'metadata.clientSecret' as it wasn't safe for managing secrets [↗](#).

To ensure safe handling of secrets, use `clientSecretFromEnv` instead.

- Removal of the deprecated `metricName` from trigger metadata section [↗](#).

The two affected Azure Scalers are **Azure Blob Scaler** and **Azure Log Analytics Scaler**. If you're using `metricName`, move `metricName` to outside the trigger metadata section to `trigger.name` in the trigger section to optionally name your trigger. The following examples show how to update configuration.

Example before KEDA 2.14

`YAML`

```
triggers:  
- type: any-type  
  metadata:  
    metricName: "my-custom-name"
```

Example by using KEDA 2.15 or 2.14

YAML

```
triggers:  
- type: any-type  
  name: "my-custom-name"  
  metadata:
```

Frequently asked questions

How can I determine whether my cluster is affected?

To determine whether your cluster is affected by the recent KEDA upgrades, follow these steps:

1. Identify the Kubernetes version of your AKS cluster.

Clusters that are running Kubernetes versions 1.30 or later will be affected by the KEDA upgrades (2.15 or 2.14). If you're on version 1.29 or earlier, these updates will not affect you. However, if you plan to upgrade your AKS cluster to Kubernetes version 1.30 or 1.31, make sure to review the changes and prepare accordingly before you upgrade.

To check the cluster version, run the following the Azure CLI command:

```
az aks show --resource-group <your-resource-group> --name <your-cluster-name> --query kubernetesVersion
```

Example of the output:

Output

```
"1.30"
```

2. Review the configurations of KEDA Scalers that are currently deployed in your cluster. Check whether Microsoft Entra pod-managed Identities are used for authentication. The following command displays output only if you're using Pod Identity together with KEDA:

```
Bash
```

```
kubectl get TriggerAuthentication --all-namespaces -o jsonpath='{range .items[?(@.spec.podIdentity.provider=="azure")]}{.metadata.namespace}{"/"}{.metadata.name}{ "\n"}{end}'
```

Example of the output:

```
Output
```

NAME	PODIDENTITY	SECRET
ENV	VAULTADDRESS	
keda-trigger-auth-azure <URL>	yourPodIdentity	azure-secret

What steps can I take to mitigate the issues?

1. Migrate from Microsoft Entra pod-managed identities to workload identity for authentication. For more information, see [Use Microsoft Entra Workload ID with AKS](#) and [Migrate from pod managed-identity to workload identity](#).
2. Update Scaler Configurations to align with the new configuration requirements:
 - For **Azure Data Explorer Scaler**, replace `metadata.clientSecret` with `clientSecretFromEnv`. For more information about the `clientSecretFromEnv` definition, see [Trigger Specifications](#).
 - For **Azure Blob Scaler** and **Azure Log Analytics Scaler**, move `trigger.metadata.metricName` to `trigger.name`.
3. Test changes in a nonproduction environment. Deploy the updated configurations in a staging or development environment to make sure that they work correctly and don't affect your applications.
4. Monitor logs and alerts. After you make changes, monitor the cluster logs for any warnings or errors that are related to the new KEDA configuration and behavior.

How can I get support if I have follow-up questions?

If you have questions or need help, [create a support request for KEDA add-on](#).

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

OrasPullNetworkTimeoutVMExtensionError error code (211) when deploying an AKS cluster

Article • 05/09/2025

This article discusses how to identify and resolve the `OrasPullNetworkTimeoutVMExtensionError` error (error code 211) that occurs when you try to create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to create an AKS cluster with the outbound type `none` or `block`, you receive the following error message:

`VMExtensionProvisioningError: VM has reported a failure when processing extension 'vmssCSE'.`

Error message: "Enable failed: failed to execute command: command terminated with exit status=211

Bootstrap Container Registry is not reachable. Please check the network configuration and try again.

Cause

For [network isolated cluster](#), egress traffic is limited. The feature introduces private Azure Container Registry (ACR) cache that acts as a proxy to download necessary binaries or images from Microsoft Artifact Registry (MAR) for AKS bootstrap. VM instances connect to the private ACR via a private link. Incorrect configuration of the private link causes VM bootstrap Custom Script Extension (CSE) to fail.

Solution

To resolve this issue, follow these steps:

1. Retrieve the ACR resource ID that AKS uses as the bootstrap ACR by running the following command:

Console

```
az aks show -g ${RESOURCE_GROUP} -n ${CLUSTER_NAME} --query  
'bootstrapProfile.containerRegistryResourceId'
```

2. Verify the ACR cache rule. It should include `aks-managed-rule` with source repo `mcr.microsoft.com/*` and target repo `aks-managed-repository/*`. Ensure no other cache rule exists with source or target repo as `*`, which override `aks-managed-rule`.
3. Review the [container registry private link](#) to ensure that the connection configuration is correct, including the private Domain Name System (DNS) zone and private link.
4. Access any failed VM instance using Secure Shell (SSH) and run curl on the ACR host. If successful, reconcile the cluster. If it still fails, return to step 3.

References

- [General troubleshooting of AKS cluster creation issues](#)
- [Network isolated Azure Kubernetes Service \(AKS\) clusters](#)
- [Container registry private link](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

OrasPullUnauthorizedVMExtensionError error code (212) when deploying an AKS cluster

Article • 05/09/2025

This article discusses how to identify and resolve the `OrasPullUnauthorizedVMExtensionError` error (error code 212) that occurs when you try to create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to create an AKS cluster with the outbound type `none` or `block`, you receive the following error message:

`VMExtensionProvisioningError: VM has reported a failure when processing extension 'vmssCSE'.`

Error message: "Enable failed: failed to execute command: command terminated with exit status=212

Bootstrap Container Registry authorization failed. Please ensure kubelet identity has pull access to the registry.

Cause

For [network isolated cluster](#), egress traffic is limited. The feature introduces private Azure Container Registry (ACR) cache that acts as a proxy to download necessary binary or images from Microsoft Artifact Registry (MAR) for AKS bootstrap. It's suggested to disable anonymous access to the ACR. The AKS node uses the kubelet identity to access the ACR. If the `acrpull` permission isn't set correctly or the kubelet identity isn't bound to the VM instance, an unauthorized error occurs.

Solution

To resolve this issue, follow these steps:

1. Access the VM instance using Secure Shell (SSH) to get the log file `/var/log/azure/cluster-provision.log`. Review the log to determine if the issue is

related to a 401 error, Azure Instance Metadata Service (IMDS) connection time-out, or an identity not found with HTTP code 400.

2. Retrieve the ACR resource ID that AKS uses as the bootstrap ACR by running the following command:

Console

```
export REGISTRY_ID=$(az aks show -g ${RESOURCE_GROUP} -n ${CLUSTER_NAME} --query 'bootstrapProfile.containerRegistryId' -o tsv)
```

3. If the issue is related to a 401 error, check if the kubelet identity has the `acrpull` permission to the ACR by running the following command:

Console

```
export KUBELET_IDENTITY_PRINCIPAL_ID=$(az aks show -g ${RESOURCE_GROUP} -n ${CLUSTER_NAME} --query 'identityProfile.kubeletIdentity.clientId' -o tsv)
```

If not, run the following command:

Console

```
az role assignment create --role AcrPull --scope ${REGISTRY_ID} --assignee-object-id ${KUBELET_IDENTITY_PRINCIPAL_ID} --assignee-principal-type ServicePrincipal
```

4. If the log error indicates that the identity isn't found, manually bind the kubelet identity to the Virtual Machine Scale Set (VMSS) for a quick fix.
5. If the issue is related to IMDS connection time-out, submit a support ticket.
6. Reconcile the cluster if the preceding operations are completed.

References

- [General troubleshooting of AKS cluster creation issues](#)
- [Network isolated Azure Kubernetes Service \(AKS\) clusters](#)
- [Container registry authentication managed identity](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the VMExtensionError_CniDownloadTimeout error code (41)

Article • 12/19/2024

This article discusses how to identify and resolve the `VMExtensionError_CniDownloadTimeout` error (also known as error code `ERR_CNI_DOWNLOAD_TIMEOUT`, error number 41) that occurs when you try to create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- The [Curl ↗](#) command-line tool

Symptoms

When you try to create a Linux-based AKS cluster, you receive the following error message:

Message: We are unable to serve this request due to an internal error

SubCode: VMExtensionError_CniDownloadTimeout;

Message="VM has reported a failure when processing extension 'vmssCSE'.

Error message: "Enable failed: failed to execute command: command terminated with exit status=41\n[stdout]\n{

"ExitCode": "41",

Cause

Your cluster nodes can't connect to the endpoint that's used to download the Container Network Interface (CNI) libraries. In most cases, this issue occurs because a network virtual appliance is blocking Secure Sockets Layer (SSL) communication or an SSL certificate.

Solution

Run a Curl command to verify that your nodes can download the binaries:

Bash

```
curl https://acs-mirror.azureedge.net/cni/azure-vnet-cni-linux-amd64-v1.0.25.tgz  
curl --fail --ssl https://acs-mirror.azureedge.net/cni/azure-vnet-cni-linux-amd64-v1.0.25.tgz --output /opt/cni/downloads/azure-vnet-cni-linux-amd64-v1.0.25.tgz
```

If you can't download these files, make sure that traffic is allowed to the downloading endpoint. For more information, see [Azure Global required FQDN/application rules](#).

References

- General troubleshooting of AKS cluster creation issues

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

Troubleshoot the VMExtensionError_OutboundConnFail error code (50)

Article • 12/30/2024

This article describes how to identify and resolve the `VMExtensionError_OutboundConnFail` error (also known as error code `ERR_OUTBOUND_CONN_FAIL`, error number 50) that might occur if you try to start or create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- The [Netcat](#) (nc) command-line tool
- The [dig](#) command-line tool
- The Client URL ([cURL](#)) tool

Symptoms

When you try to start or create an AKS cluster, you receive the following error message:

Unable to establish outbound connection from agents, please see <https://aka.ms/aks-required-ports-and-addresses> for more information.

Details: Code="VMExtensionProvisioningError"

Message="VM has reported a failure when processing extension 'vmssCSE'.

Error message: "Enable failed: failed to execute command: command terminated with exit status=50\n[stdout]\n\n[stderr]\\nn: connect to mcr.microsoft.com port 443 (tcp) failed: Connection timed out\\nCommand exited with non-zero status

Error details : "vmssCSE error messages : {vmssCSE exit status=50, output=pt/apt.conf.d/95proxy...}

Cause

The custom script extension that downloads the necessary components to provision the nodes couldn't establish the necessary outbound connectivity to obtain packages. For public clusters, the nodes try to communicate with the Microsoft Container Registry (MCR) endpoint (`mcr.microsoft.com`) on port 443.

There are many reasons why the traffic might be blocked. In any of these situations, the best way to test connectivity is to use the Secure Shell protocol (SSH) to connect to the node. To make the connection, follow the instructions in [Connect to Azure Kubernetes Service \(AKS\) cluster nodes for maintenance or troubleshooting](#). Then, test the connectivity on the cluster by following these steps:

1. After you connect to the node, run the `nc` and `dig` commands:

Bash

```
nc -vz mcr.microsoft.com 443
dig mcr.microsoft.com 443
```

ⓘ Note

If you can't access the node through SSH, you can test the outbound connectivity by running the `az vmss run-command invoke` command against the Virtual Machine Scale Set instance:

Azure CLI

```
# Get the VMSS instance IDs.
az vmss list-instances --resource-group <mc-resource-group-name> \
    --name <vmss-name> \
    --output table

# Use an instance ID to test outbound connectivity.
az vmss run-command invoke --resource-group <mc-resource-group-
    name> \
    --name <vmss-name> \
    --command-id RunShellScript \
    --instance-id <vmss-instance-id> \
    --output json \
    --scripts "nc -vz mcr.microsoft.com 443"
```

2. If you try to create an AKS cluster by using an HTTP proxy, run the `nc`, `curl`, and `dig` commands after you connect to the node:

Bash

```
# Test connectivity to the HTTP proxy server from the AKS node.  
nc -vz <http-s-proxy-address> <port>  
  
# Test traffic from the HTTP proxy server to HTTPS.  
curl --proxy http://<http-proxy-address>:<port>/ --head  
https://mcr.microsoft.com  
  
# Test traffic from the HTTPS proxy server to HTTPS.  
curl --proxy https://<https-proxy-address>:<port>/ --head  
https://mcr.microsoft.com  
  
# Test DNS functionality.  
dig mcr.microsoft.com 443
```

ⓘ Note

If you can't access the node through SSH, you can test the outbound connectivity by running the `az vmss run-command invoke` command against the Virtual Machine Scale Set instance:

Azure CLI

```
# Get the VMSS instance IDs.  
az vmss list-instances --resource-group <mc-resource-group-name> \  
--name <vmss-name> \  
--output table  
  
# Use an instance ID to test connectivity from the HTTP proxy  
server to HTTPS.  
az vmss run-command invoke --resource-group <mc-resource-group-  
name> \  
--name <vmss-name> \  
--command-id RunShellScript \  
--instance-id <vmss-instance-id> \  
--output json \  
--scripts "curl --proxy http://<http-proxy-address>:<port>/ --  
head https://mcr.microsoft.com"  
  
# Use an instance ID to test connectivity from the HTTPS proxy  
server to HTTPS.  
az vmss run-command invoke --resource-group <mc-resource-group-  
name> \  
--name <vmss-name> \  
--command-id RunShellScript \  
--instance-id <vmss-instance-id> \  
--output json \  
--scripts "curl --proxy https://<https-proxy-address>:<port>/ -  
-head https://mcr.microsoft.com"  
  
# Use an instance ID to test DNS functionality.
```

```
az vmss run-command invoke --resource-group <mc-resource-group-name> \
    --name <vmss-name> \
    --command-id RunShellScript \
    --instance-id <vmss-instance-id> \
    --output json \
    --scripts "dig mcr.microsoft.com 443"
```

Solution

The following table lists specific reasons why traffic might be blocked, and the corresponding solution for each reason:

[+] Expand table

Issue	Solution
Traffic is blocked by firewall rules, a proxy server or Network Security Group (NSG)	This issue occurs when the AKS required ports or Fully Qualified Domain Names (FQDNs) are blocked by a firewall, proxy server, or NSG. Ensure these ports and FQDNs are allowed. To determine what's blocked, check the connectivity provided in the preceding Cause section. For more information about AKS required ports and FQDNs, see Outbound network and FQDN rules for Azure Kubernetes Service (AKS) clusters .
The AAAA (IPv6) record is blocked on the firewall	On your firewall, verify that nothing exists that would block the endpoint from resolving in Azure DNS.
Private cluster can't resolve internal Azure resources	In private clusters, the Azure DNS IP address (168.63.129.16) must be added as an upstream DNS server if custom DNS is used. Verify that the address is set on your DNS servers. For more information, see Create a private AKS cluster and What is IP address 168.63.129.16?

More information

- General troubleshooting of AKS cluster creation issues

Third-party contact disclaimer

Microsoft provides third-party contact information to help you find additional information about this topic. This contact information may change without notice. Microsoft does not guarantee the accuracy of third-party contact information.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

Troubleshoot the VMExtensionError_K8SAPIServerConnFail error code (51)

Article • 12/19/2024

This article discusses how to identify and resolve the `VMExtensionError_K8SAPIServerConnFail` error (also known as error code `ERR_K8S_API_SERVER_CONN_FAIL`, error number 51) that occurs when you try to start or create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- The [Netcat](#) (nc) command-line tool

Symptoms

When you try to start or create an AKS cluster, you receive the following error message:

Unable to establish connection from agents to Kubernetes API server, please see <https://aka.ms/aks-required-ports-and-addresses> for more information.

Details: Code="VMExtensionProvisioningError"

Message="VM has reported a failure when processing extension 'vmssCSE'.

Error message: "Enable failed: failed to execute command: command terminated with exit status=51\n[stdout]\n{

"ExitCode": "51",

"Output": "Thu Oct 14 18:07:37 UTC 2021,aks-nodepool1-18315663-vmss000000\nConnection to

Cause

Your cluster nodes can't connect to your cluster API server pod.

Solution

Run a Netcat command to verify that your nodes can resolve the cluster's fully qualified domain name (FQDN):

```
shell
```

```
nc -vz <cluster-fqdn> 443
```

If you're using egress filtering through a firewall, make sure that traffic is allowed to your cluster FQDN.

In rare cases, the firewall's outbound IP address can be blocked if you've authorized IP addresses that are enabled on your cluster. In this scenario, you must add the outbound IP address of your firewall to the list of authorized IP ranges for the cluster. For more information, see [Secure access to the API server using authorized IP address ranges in AKS](#).

More information

- [General troubleshooting of AKS cluster creation issues](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

Troubleshoot the VMExtensionError_K8SAPIServerDNSLookupFail error code (52)

Article • 03/12/2025

This article discusses how to identify and resolve the `VMExtensionError_K8SAPIServerDNSLookupFail` error (also known as error code `ERR_K8S_API_SERVER_DNS_LOOKUP_FAIL`, error number 52) that occurs when you try to start or create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- The `nslookup` DNS lookup tool for Windows nodes or the `dig` [tool](#) for Linux nodes.
- [Azure CLI](#), version 2.0.59 or a later version. If Azure CLI is already installed, you can find the version number by running `az --version`.

Symptoms

When you try to start or create an AKS cluster, you receive the following error message:

Agents are unable to resolve Kubernetes API server name. It's likely custom DNS server is not correctly configured, please see <https://aka.ms/aks/private-cluster#hub-and-spoke-with-custom-dns> for more information.

Details: Code="VMExtensionProvisioningError"

Message="VM has reported a failure when processing extension 'vmssCSE'.

Error message: "Enable failed: failed to execute command: command terminated with exit status=52\n[stdout]\n{

"ExitCode": "52",

"Output": "Fri Oct 15 10:06:00 UTC 2021,aks- nodepool1-36696444-vmss000000\nConnection to mcr.microsoft.com 443 port [tcp/https]

Cause

The cluster nodes can't resolve the cluster's fully qualified domain name (FQDN) in Azure DNS. Run the following DNS lookup command on the failed cluster node to find DNS resolutions that are valid.

[+] Expand table

Node OS	Command
Linux	<code>dig <cluster-fqdn></code>
Windows	<code>nslookup <cluster-fqdn></code>

Solution

On your DNS servers and firewall, make sure that nothing blocks the resolution to your cluster's FQDN. Your custom DNS server might be incorrectly configured if something is blocking even after you run the `nslookup` or `dig` command and apply any necessary fixes. For help to configure your custom DNS server, review the following articles:

- [Create a private AKS cluster](#)
- [Private Azure Kubernetes service with custom DNS server ↗](#)
- [What is IP address 168.63.129.16?](#)

When you use a private cluster that has a custom DNS, a DNS zone is created. The DNS zone must be linked to the virtual network. This occurs after the cluster is created. Creating a private cluster that has a custom DNS fails during creation. However, you can restore the creation process to a "success" state by reconciling the cluster. To do this, run the `az resource update` command in Azure CLI, as follows:

```
Azure CLI

az resource update --resource-group <resource-group-name> \
    --name <cluster-name> \
    --namespace Microsoft.ContainerService \
    --resource-type ManagedClusters
```

Also verify that your DNS server is configured correctly for your private cluster, as described earlier.

 **Note**

Conditional Forwarding doesn't support subdomains.

More information

- General troubleshooting of AKS cluster creation issues

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

VMExtensionError_VHDFileNotFound error code (65) when deploying an AKS cluster

Article • 02/25/2025

This article discusses how to identify and resolve the `VMExtensionError_VHDFileNotFound` error code (error code number 65) that occurs when you try to create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to create an AKS cluster, you receive the following error message:

`VMExtensionProvisioningError: VM has reported a failure when processing extension 'vmssCSE'.`

Error message: "Enable failed: failed to execute command: command terminated with exit status=65

Cause

Under rare circumstances, the 65 exit code for the Azure Virtual Machine Scale Set custom script extension (`vmssCSE`) might happen instead of the following error codes:

[+] Expand table

Error code name	Error code number
<code>OutboundConnFailVMExtensionError</code>	50
<code>K8SAPIServerConnFailVMExtensionError</code>	51
<code>K8SAPIServerDNSLookupFailVMExtensionError</code>	52

This error occurs if a connectivity issue exists between your AKS cluster and the required Azure endpoints, such as `mcr.microsoft.com` or `acs-mirror.azureedge.net`.

Solution

Review [outbound network and FQDN rules for Azure Kubernetes Service \(AKS\) clusters](#) and make sure that all API services FQDN are allowed.

For detailed troubleshooting steps, refer to the troubleshooting guides in the following articles:

- Troubleshoot the VMExtensionError_OutboundConnFail error code (50)
- Troubleshoot the VMExtensionError_K8SAPIServerConnFail error code (51)
- Troubleshoot the VMExtensionError_K8SAPIServerDNSLookupFail error code (52)

References

- General troubleshooting of AKS cluster creation issues
- Outbound network and FQDN rules for Azure Kubernetes Service (AKS) clusters

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

Troubleshoot the VMExtensionProvisioningTimeout error code

Article • 04/16/2025

This article discusses how to identify and resolve the `VMExtensionProvisioningTimeout` error that occurs when you try to create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- [Azure CLI](#), version 2.28.0 or a later version. If Azure CLI is already installed, you can find the version number by running `az --version`.

Symptoms

When you try to create an AKS cluster by using the Azure CLI, you receive the "VMExtensionProvisioningTimeout" error with text like the following example:

Output

```
Failed to reconcile agent pool agentpool0: err: VMSSAgentPoolReconciler retry failed:  
Category: InternalError;  
SubCode: VMExtensionProvisioningTimeout;  
Dependency: Microsoft.Compute/VirtualMachineScaleSet;  
OrginalError:  
Code="VMExtensionProvisioningTimeout"  
Message="Provisioning of VM extension vmssCSE has timed out. Extension provisioning has taken too long to complete. The extension last reported \"Plugin enabled\".\r\n\r\nMore information on troubleshooting is available at <https://aka.ms/VMExtensionCSELinuxTroubleshoot>";  
AKSTeam: NodeProvisioning,  
Retriable: true
```

You also can [view the error details in the Azure portal](#).

Cause

Several different issues can cause the "VMExtensionProvisioningError" class of errors. However, the troubleshooting steps are the same for all the issues. Possible causes are as follows:

- The custom script extension that provisions the virtual machines (VMs) can't establish a connection to the endpoint that's used for downloading the Kubernetes binaries.
- The custom script extension that provisions the VMs can't establish a connection to the endpoint that's used for downloading the CNI binaries.
- The custom script extension that provisions the VMs can't establish the required outbound connectivity to obtain packages.
- The cluster can't resolve the necessary Domain Name System (DNS) address to correctly provision the node.
- The custom script extension that provisions the VMs reached a timeout while running a packet management update (such as [apt-get](#) in case the node pool uses Linux).

Solution

Follow these steps:

1. If egress filtering is set up on the cluster (such as [custom user-defined routes](#)), see [Limit network traffic with Azure Firewall in Azure Kubernetes Service \(AKS\)](#) and [Outbound network and FQDN rules for AKS clusters](#) to view the necessary prerequisites, and make sure that your setup meets the prerequisites.
2. On your DNS servers and firewall, make sure that nothing blocks the resolution of your cluster's fully qualified domain name (FQDN).
3. Because your custom DNS server might be configured incorrectly, review the following articles if FQDN resolution continues to be blocked:
 - [Create a private AKS cluster](#)
 - [Private Azure Kubernetes service with custom DNS server](#)
 - [What is IP address 168.63.129.16?](#)

More information

- [General troubleshooting of AKS cluster creation issues](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the AvailabilityZoneNotSupportedException error code

06/04/2025

This article discusses how to identify and resolve the "AvailabilityZoneNotSupportedException" error that occurs when you try to create an Azure Kubernetes Service (AKS) cluster.

Prerequisites

Access to [Azure CLI](#).

Symptoms

An AKS cluster creation fails in specified availability zones, and you receive an "AvailabilityZoneNotSupportedException" error with the following message:

Preflight validation check for resource(s) for container service <resource-name> in resource group <resource-group-name> failed. Message: The zone(s) '1' for resource '<agentpoolName>' is not supported. The supported zones for location '<location>' are 'A', 'B'

Cause

The issue occurs because the requested SKU has restrictions in some or all zones of your subscription. To verify the restrictions, go to the [Verify SKU restrictions](#) section.

Solution

To resolve this issue, follow the [Azure region access request process](#) to request access to the specified region or zone.

Verify SKU restrictions

1. List the SKU details by running one of the following commands:

Azure CLI

```
az vm list-skus -l <location> --size <SKU>
```

Azure CLI

```
az rest --method get \
    --url
"https://management.azure.com/subscriptions/<subscription>/providers/Microsoft.Compute/skus?%24filter=location+eq+%27<location>%27&api-version=2022-03-01"
>> availableSkus.txt
```

ⓘ Note

Replace <subscription>, <SKU>, and <location> accordingly.

2. Search for the requested SKU from the command output.

3. If you see information like the following, it indicates that the requested SKU has restrictions in some or all zones of your subscription:

JSON

```
"restrictions": [
    {
        "type": "Zone",
        "values": [
            "<zone>"
        ],
        "restrictionInfo": {
            "locations": [
                "<location>"
            ],
            "zones": [
                "1",
                "2",
                "3"
            ]
        },
        "reasonCode": "NotAvailableForSubscription"
    }
]
```

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the AksCapacityHeavyUsage error code

06/27/2025

This article discusses how to identify and resolve the `AksCapacityHeavyUsage` error that might occur when you create a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to create an AKS cluster, you receive the following error message:

Code: AksCapacityHeavyUsage

Message: AKS is experiencing heavy usage in region <Region>. We are working on adding new capacity. In the meantime, please consider creating new AKS clusters in a different region. For a list of all the Azure regions, visit <https://aka.ms/aks/regions>. For more details on this error, visit <https://aka.ms/akscapacityheavyusage>.

Cause

You're trying to create a cluster in a region that has limited capacity.

When you create an AKS cluster, Microsoft Azure allocates compute resources to your subscription. You might occasionally experience the `AksCapacityHeavyUsage` error because of significant growth in demand for Azure Kubernetes Service in specific regions.

Resolution

Solution 1: Select a different region

The easiest and quickest solution is to try to deploy to a different region (for example, NorthEurope instead of WestEurope or UAENorth instead of QatarCentral). To find nearby regions, visit the [Azure Geographies page](#).

This approach might not be feasible if you already have existing resources in the requested region, but it's the preferred solution in a dev/test scenario.

Solution 2: Deploy a cluster that has different settings

The infrastructure that hosts AKS-managed clusters have different allocation reservations. Therefore, AKS might have more capacity for public clusters than it has for private clusters. If you experience the `AksCapacityHeavyUsage` error when you try to create a private cluster, try to create a public cluster instead (or vice versa).

Solution 3: Use an Azure Enterprise subscription

When capacity is running low, non-Enterprise Agreement (EA) subscriptions are limited first in AKS cluster creation to reserve resources for real production scenarios. If you have an EA subscription, make sure that you use the EA subscription to create the AKS cluster.

Solution 4: Retry the operation

Capacity is often reclaimed when other users stop or delete their AKS clusters. Therefore, the operation might succeed if you retry it later.

More information

- Ensuring capacity for users is a top priority for Microsoft, and we're working to scale up our infrastructure to accommodate the increasing popularity of Azure services.

For more information about improvements that we're making toward delivering a resilient cloud supply chain, see [this September 2021 Azure Blog article](#).

- [General troubleshooting of AKS cluster creation issues](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

AKSOperationPreempted or AKSOperationPreemptedByDelete error when performing a new operation

Article • 04/16/2025

This article discusses how to identify and resolve the `AKSOperationPreempted` or `AKSOperationPreemptedByDelete` error that might occur when you try to perform a new operation but it has been preempted by another operation on a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to perform a new operation on an AKS cluster, you receive one of the following error messages:

- Code: "AKSOperationPreempted"
Message: "This operation has been preempted by another operation with ID <operation ID>"

This message indicates that the operation has been preempted by another operation, and the GUID for the new operation is given in the error message.

- Code: "AKSOperationPreemptedByDelete"
Message: "This operation has been preempted by Deleting operation."

This message indicates that the operation has been preempted by a delete operation.

Cause

This error usually occurs when an in-progress operation is interrupted by a subsequent operation that was issued before the in-progress operation is finished. The error will indicate the subsequent operation, which can be a delete or any other operation.

Solution

To resolve this issue, use one of the following methods. Once there are no operations running, you can try to run your operation again.

- Wait for the previous operation to complete.

You can check the status of the operation using the ID given in the error message with the following command:

```
Azure CLI
```

```
az aks operation show \
--resource-group myResourceGroup \
--name myCluster \
--operation-id "<operation-id>"
```

- Run an `abort` command to stop the previous operation.

For more information about how to abort an operation, see [Terminate a long running operation on an Azure Kubernetes Service \(AKS\) cluster](#).

More information

[General troubleshooting of AKS cluster creation issues](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Error AADSTS7000222 - BadRequest or InvalidClientSecret

Article • 05/19/2025

This article discusses how to identify and resolve the `AADSTS7000222` error (`BadRequest` or `InvalidClientSecret`) that occurs when you try to create or upgrade a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- [Azure CLI](#)

Symptoms

When you try to create or upgrade an AKS cluster, you receive one of the following error messages.

 Expand table

Error code	Message
<code>BadRequest</code>	The credentials in <code>ServicePrincipalProfile</code> were invalid. Please see https://aka.ms/aks-sp-help for more details. (Details: adal: Refresh request failed. Status Code = '401'. Response body: {"error": "invalid_client", "error_description": "AADSTS7000222: The provided client secret keys for app '<application-id>' are expired. Visit the Azure portal to create new keys for your app: https://aka.ms/NewClientSecret , or consider using certificate credentials for added security: https://aka.ms/certCreds ."}"
<code>InvalidClientSecret</code>	Customer auth is not valid for tenant: <tenant-id>: adal: Refresh request failed. Status Code = '401'. Response body: {"error": "invalid_client", "error_description": "AADSTS7000222: The provided client secret keys for app '<application-id>' are expired. Visit the Azure portal to create new keys for your app: https://aka.ms/NewClientSecret , or consider using certificate credentials for added security: https://aka.ms/certCreds ."}"

Cause

The issue that generates this service principal alert usually occurs for one of the following reasons:

- The client secret expired.

- Incorrect credentials were provided.
- The service principal doesn't exist within the Microsoft Entra ID tenant of the subscription.

Verify the cause

Run the following Azure CLI code to retrieve the service principal profile for your AKS cluster and [check the expiration date of the service principal](#):

Azure CLI

```
SP_ID=$(az aks show --resource-group <rg-name> \
    --name <aks-cluster-name> \
    --query servicePrincipalProfile.clientId \
    --output tsv)
az ad app credential list --id "$SP_ID"
```

Alternatively, you can verify that the service principal name and secret are correct and aren't expired. To do this, follow these steps:

1. In the [Azure portal](#), search for and select **Microsoft Entra ID**.
2. In the navigation pane of Microsoft Entra ID, select **App registrations**.
3. On the **Owned applications** tab, select the affected application.
4. Find the service principal name and secret information, and verify that the information is correct and current.

Solution

1. In the [Update or rotate the credentials for an AKS cluster](#) article, follow the instructions in one of the following article sections, as appropriate:
 - [Reset the existing service principal credentials](#)
 - [Create a new service principal](#)
2. Using your new service principal credentials, follow the instructions in the [Update AKS cluster with service principal credentials](#) section of that article.

More information

- [Use a service principal with Azure Kubernetes Service \(AKS\)](#) (especially the [Troubleshoot](#) section)

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

A load balancer with a private link service or a private link service with private endpoint connections cannot be deleted

Article • 05/19/2025

This article discusses how to identify and resolve the

`CannotDeleteLoadBalancerWithPrivateLinkService` or

`PrivateLinkServiceWithPrivateEndpointConnectionsCannotBeDeleted` error that occurs when you try to delete a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to delete an AKS cluster, you receive one of the following error messages:

- For the `CannotDeleteLoadBalancerWithPrivateLinkService` error code:

```
internalErrorCode: "CannotDeleteLoadBalancerWithPrivateLinkService"
```

```
StatusCode=BadRequest
```

```
{
```

```
"Cannot delete load balancer
```

```
...../providers/Microsoft.Network/loadBalancers/kubernetes-internal since it is  
referenced by private link service
```

```
...../providers/Microsoft.Network/privateLinkServices/test. Please delete private link  
service prior to deleting load balancer."
```

```
}
```

- For the `PrivateLinkServiceWithPrivateEndpointConnectionsCannotBeDeleted` error code:

```
internalErrorCode:
```

```
"PrivateLinkServiceWithPrivateEndpointConnectionsCannotBeDeleted"
```

```
StatusCode=BadRequest
```

```
{
```

```
message: "Private link service ...../Microsoft.Network/privateLinkServices/test cannot  
be deleted since it has 1 private endpoint connection. Please delete all private  
endpoint connections before deleting the private link service."
```

Cause

The private link service can't be deleted because the cluster is associated with private endpoint connections.

Solution

Make sure that the private link service isn't associated with any private endpoint connections. Delete all private endpoint connections before you delete the private link service. For more information, see [Azure: How to delete a private link service that has a private endpoint connected to it?](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

CreateOrUpdateVirtualNetworkLinkFailed error when updating or upgrading an AKS cluster

Article • 05/19/2025

This article provides a solution to the "CreateOrUpdateVirtualNetworkLinkFailed" error code that occurs when you try to update or upgrade a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

[Azure CLI](#)

Symptoms

An AKS cluster update or upgrade operation fails and returns the following error message:

Code: CreateOrUpdateVirtualNetworkLinkFailed - SubCode: BadRequest

Message: Reconcile private dns failed

Details: Create or update virtual network link failed. Subscription: <SubscriptionID>; resource group: <RGName>; private dns zone: <GUID>.privatelink.<region>.azmk8s.io; virtual network link: <VNET_Link>.

Message: A virtual network cannot be linked to multiple zones with overlapping namespaces. You tried to link virtual network with '<GUID>.privatelink.<region>.azmk8s.io' and '<GUID>.privatelink.<region>.azmk8s.io' zones.

Cause

This error happens in this scenario:

- You disassociate the original private Domain Name System (DNS) zone of the AKS cluster.
- You link a private DNS zone that has the same name as the original zone but is located in a different resource group or subscription.

That's why you see the same private DNS zone name "<GUID>.privatelink.<region>.azmk8s.io" in the error message. The first is the new zone in the new resource group or subscription, while the second is the original zone created with the AKS cluster.

Solution

To resolve this issue, follow these steps:

1. Remove the link between the AKS cluster's virtual network (VNET) and the private DNS zone created in the wrong resource group or subscription.
2. Update the cluster by running the following command:

Azure CLI

```
az aks update -n <myAKSCluster> -g <myResourceGroup>
```

The command output should show the cluster's `ProvisioningState` as `Running`.

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the CustomPrivateDNSZoneMissingPermission Error error code

Article • 05/19/2025

This article discusses how to identify and resolve the "CustomPrivateDNSZoneMissingPermissionError" error code that occurs when you try to create or update a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- [Azure CLI](#), version 2.53.0 or a later version. To find your installed version, run `az --version`.

Symptoms

An AKS cluster create or update operation fails and returns the following error message:

Code: CustomPrivateDNSZoneMissingPermissionError

Message: Service principal or user-assigned identity must be given permission to read and write to custom private dns zone <custom-private-dns-zone-resource-id>. Check access result not allowed for action Microsoft.Network/privateDnsZones/read.

Cause

Before AKS runs a cluster create or update operation for a private cluster that uses a [custom private DNS zone](#), it checks whether the cluster's managed identity or service principal has the required permissions to control the private DNS zone. If AKS doesn't find the necessary permissions, it blocks the operation so that the cluster doesn't enter a failed state.

Solution

To create the missing role assignment, follow these steps:

1. Get the resource ID of the cluster's private DNS zone by running the [az aks show](#) command, and store it as the `CUSTOM_PRIVATE_DNS_ZONE_ID` variable:

Azure CLI

```
CUSTOM_PRIVATE_DNS_ZONE_ID=$(az aks show \
--resource-group <aks-resource-group> \
--name <aks-cluster-name> \
--query apiServerAccessProfile.privateDnsZone \
--output tsv)
```

ⓘ Note

Because the resource ID of the custom private DNS zone was also shown in the original error message, you can alternatively assign that resource ID to the variable instead of running the `az aks show` command.

2. Assign the [Private DNS Zone Contributor](#) role to the cluster's managed identity or service principal by running the `az role assignment create` command:

Azure CLI

```
az role assignment create --role "Private DNS Zone Contributor" \
--scope $CUSTOM_PRIVATE_DNS_ZONE_ID \
--assignee <control-plane-principal-id>
```

ⓘ Note

It can take up to 60 minutes to finish granting permissions to your cluster's managed identity or service principal.

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the DnsServiceIpOutOfServiceCidr error code

Article • 05/19/2025

This article discusses how to identify and resolve the `DnsServiceIpOutOfServiceCidr` error that occurs when you try to create or upgrade an Azure Kubernetes Service (AKS) cluster.

Prerequisites

This article requires Azure CLI version 2.0.65 or a later version. To find the version number, run `az --version`. If you have to install or upgrade Azure CLI, see [How to install the Azure CLI](#).

For more detailed information about the upgrade process, see [Upgrade an AKS cluster](#).

Symptoms

An AKS cluster create operation fails, and you receive a `DnsServiceIpOutOfServiceCidr` error message.

```
(DnsServiceIpOutOfServiceCidr) The DNS service IP 10.0.0.10 is out of the range defined by service CIDR 10.200.0.0/16.
```

- **Code:** `DnsServiceIpOutOfServiceCidr`
- **Message:** The DNS service IP 10.0.0.10 is out of the range defined by service CIDR 10.200.0.0/16.
- **Target:** networkProfile.dnsServiceIP

Cause

This error occurs if the DNS service IP (`--dns-service-ip`) for AKS is out of the range that's defined by the service CIDR (`--service-cidr`).

Solution

Make sure that the DNS service IP is within the range that's defined by service CIDR.

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Troubleshoot InUseRouteTableCannotBeDeleted error code

Article • 05/19/2025

This article discusses how to identify and resolve the `InUseRouteTableCannotBeDeleted` error that occurs when you try to delete a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to delete an AKS cluster, you receive the following error message:

```
Error Code: "InUseRouteTableCannotBeDeleted"  
{  
    "Route table aks-agentpool-routetable is in use and cannot be deleted.  
    ..../providers/Microsoft.Network/routeTables/aks-agentpool-test-routetable"  
}
```

Cause

You tried to delete the AKS cluster while its associated route table was still in use.

Solution

Remove the associated route table from the subnet. For instructions, see [Dissociate a route table from a subnet](#). Then, try again to delete the AKS cluster.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the InvalidParameter error

Article • 04/16/2025

This article discusses how to identify and resolve the `InvalidParameter` error that occurs when you try to create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- [Azure CLI](#), version 2.0.81 or a later version. If Azure CLI is already installed, you can find the version number by running `az --version`.

Symptoms

When you create an AKS cluster, the provided configurations are usually validated before the cluster is created. However, on rare occasions, a parameter passes validation before the AKS cluster is created but causes errors when the resources for the cluster are created. Errors that are related to invalid parameters might resemble the following examples:

- Scenario: The selected VM size is not available

```
Code="InvalidParameter"

Message="**The requested VM size Standard_D4s_v3 is not available in the
current region. The sizes available in the current region are:
ExtraSmall_Internal, Small_Internal, Medium_Internal, Large_Internal,
ExtraLarge_Internal, Standard_DC2as_v5, Standard_DC4as_v5, Standard_DC8as_v5,
Standard_DC16as_v5, Standard_DC32as_v5, Standard_DC48as_v5,
Standard_DC64as_v5, Standard_DC96as_v5, Standard_DC2ads_v5,
Standard_DC4ads_v5, Standard_DC8ads_v5, Standard_DC16ads_v5,
Standard_DC32ads_v5, Standard_DC48ads_v5, Standard_DC64ads_v5,
Standard_DC96ads_v5, Standard_EC2as_v5, Standard_EC4as_v5, Standard_EC8as_v5,
Standard_EC16as_v5, Standard_EC20as_v5, Standard_EC32as_v5,
Standard_EC48as_v5, Standard_EC64as_v5, Standard_EC96as_v5,
Standard_EC96ias_v5, Standard_EC2ads_v5, Standard_EC4ads_v5,
Standard_EC8ads_v5, Standard_EC16ads_v5, Standard_EC20ads_v5,
Standard_EC32ads_v5, Standard_EC48ads_v5, Standard_EC64ads_v5,
Standard_EC96ads_v5, Standard_EC96iads_v5.\r\nFind out more on the available
VM sizes in each region at <https://aka.ms/azureregions>.""

Target="vmSize"
```

- Scenario: Cluster names are unavailable or conflict with Azure reserved values
 - Example 1

```
Code="InvalidParameter"

Message="The value of parameter name is invalid. Error details: "omsagent-aks-dev-microsoft" managed cluster name is invalid because 'MICROSOFT' and 'WINDOWS' can't be used as either a whole word or a substring in the name.. Please see https://aka.ms/aks-naming-rules for more details."
```

- o Example 2

```
Message="The value of parameter name is invalid. Error details: "login" managed cluster name is invalid because 'LOGIN' and 'XBOX' can't be used at the start of a resource name, but can be used later in the name.. Please see https://aka.ms/aks-naming-rules for more details."
```

- o Example 3

```
Message=" The value of parameter name is invalid. Error details: "azure" managed cluster name is invalid because it is reserved.. Please see https://aka.ms/aks-naming-rules for more details.
Target: name"
```

Cause

This issue occurs because one of the following conditions is true:

- The Azure Virtual Machine SKU isn't available in the selected region.
- The service principal is invalid.
- A virtual network, subnet, or route table is invalid.
- An Azure CLI parameter is invalid.
- The value of parameter name is unavailable or reserved by Azure.

There might also be other reasons that your cluster creation attempt failed.

Solution

In the following table, follow the link for the appropriate troubleshooting step.

Troubleshooting step	Reference link
Check whether the SKU is available	Resolve errors for SKU not available
Verify that the service principal is valid	Service principals together with AKS
Verify that any commands that were used to create the cluster are valid	az aks (Azure CLI reference)
Verify that any custom network resources that were used to create the cluster are valid	Configure Azure CNI networking in AKS and Customize cluster egress with a user-defined route
Avoid using unavailable or Azure-reserved values for names	Refer to the error messages provided

More information

- [General troubleshooting of AKS cluster creation issues](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the InvalidResourceReference error code

Article • 04/16/2025

This article discusses how to identify and resolve the `InvalidResourceReference` error that may occur when you try to create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster or update an AKS cluster.

Symptom 1

When you try to create an AKS cluster, you receive the following error message:

```
Code="InvalidResourceReference"

Message="Resource
/subscriptions/<subscription-id-
guid>/resourceGroups/MyResourceGroup/providers/Microsoft.Network/virtualNetwork
s/vnet-otcom/subnets/Subnet-AKS
referenced by resource
/subscriptions/<subscription-id-
guid>/resourceGroups/MC_MyResourceGroup_MyCluster-
AKS_JAPANEAST/providers/Microsoft.Compute/virtualMachineScaleSets/aks-nodepool-
vmss
was not found. Please make sure that the referenced resource exists, and that both
resources are in the same region."
Details=[]
```

Cause 1

Here are the possible causes of this issue:

- A mismatch exists between resources in different regions.

The example in [Symptom 1](#) shows that the virtual network and the virtual machine scale set aren't in the same region. Because the resources are in different regions, it's impossible to create the scale set instance.

- The referenced resource has been manually modified or deleted.

Solution 1

If a mismatch exists between resources in different regions, review the resources to make sure that they're in the same region. In this example, either modify the region where the AKS cluster is being built, or create a new virtual network in the same region.

If the referenced resource has been manually modified or deleted, it might be difficult to resolve this issue because it's unsupported to manually modify the underlying IaaS resources in the *MC_* resource group. A possible solution might be to recreate the deleted resource, reassociate it with the VMSS, and then trigger an update on the AKS cluster. However, as this is an unsupported scenario, the success of this solution can't be guaranteed.

Symptom 2

When you try to update an AKS cluster, you receive the following error message:

```
Code="InvalidResourceReference"
Message="Resource
/subscriptions/<subscription-id-
guid>/resourceGroups/MC_MyResourceGroup/providers/Microsoft.Network/loadBalancers
/kubernetes/frontendIPConfigurations/<frontendIP_ID> referenced by resource
/subscriptions/<subscription-id-
guid>/resourceGroups/MC_MyResourceGroup/providers/Microsoft.Network/loadBalancers
/kubernetes/loadBalancingRules/<frontend_IP_rule> was not found. Please make sure
that the referenced resource exists, and that both resources are in the same region."
Message="Resource
Details=[]
```

Cause 2

This issue might occur if the default outbound rule "aksOutboundRule" on the load balancer is manually modified. This unexpected modification typically occurs when the outbound IP is updated if you update the cluster without the `load-balancer-outbound-ips` parameter.

Solution 2

Rerun the `az aks update` command with the `load-balancer-outbound-ips` parameter to update your cluster. Use the resource ID of the public IP as the parameter value. For more information, see [Update the cluster with your own outbound public IP](#).

More information

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

InvalidLoadBalancerProfileAllocatedOutboundPorts error when creating or updating an AKS cluster

Article • 05/19/2025

This article discusses how to identify and resolve the "InvalidLoadBalancerProfileAllocatedOutboundPorts" error code that occurs when you try to create or update a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

Azure CLI, version 2.53.0 or a later version. To find your installed version, run the `az --version` command.

Symptoms

An AKS cluster create or update operation fails and returns the following error message:

Code: InvalidLoadBalancerProfileAllocatedOutboundPorts

Message: Load balancer profile allocated ports 8000 is not in an allowable range given the number of nodes and IPs provisioned. Total node count 9 requires 72000 ports but only 64000 ports are available given 1 outbound public IPs. Refer to <https://aka.ms/aks/slbp-ports> for more details.

Cause

In clusters with outbound type `LoadBalancer`, the traffic originating from cluster nodes (including traffic from applications running in pods) egresses via the AKS-managed load balancer, which relies on Source Network Address Translation (SNAT) ports to establish these outbound connections.

By default, each cluster is assigned one outbound frontend IP address that allows for a total of 64,000 SNAT ports, and each worker node in the cluster is allocated a share of these ports. For example, if a cluster has 50 nodes or fewer, each worker node is allocated 1,024 ports. In some scenarios where applications running in a cluster require establishing large numbers of outbound connections, the default number of available SNAT ports may be insufficient, which leads to SNAT port exhaustion.

To solve the SNAT port exhaustion issue, use one or both of the following methods:

- Change the number of outbound frontend IP addresses. Each IP address provides an additional 64,000 SNAT ports.
- Change the number of SNAT ports assigned to each worker node.

The "InvalidLoadBalancerProfileAllocatedOutboundPorts" error occurs when the specified node count, the number of outbound frontend IP addresses, and the number of allocated ports per node don't constitute a feasible configuration.

Solution

To resolve the "InvalidLoadBalancerProfileAllocatedOutboundPorts" error, follow these steps:

1. Use the following formula to check if the desired configuration is feasible:

code

```
64,000 ports per IP / <outbound ports per node> * <number of outbound IPs> =  
<maximum number of nodes in the cluster>
```

 **Note**

When performing this check, make sure you consider node surges that happen during cluster upgrades and other operations. AKS defaults to one buffer node for upgrade operations, but this number can be modified using the [maxSurge](#) parameter.

2. Change the cluster's node count, the number of outbound frontend IP addresses, or the number of SNAT ports per node.

For more information about how to configure the allocated outbound ports and examples and calculations of the number of ports you might need, see [Configure the allocated outbound ports](#).

Reference

[Use Source Network Address Translation \(SNAT\) for outbound connections](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the LinkedAuthorizationFailed error code

Article • 04/16/2025

This article discusses how to identify and resolve the `LinkedAuthorizationFailed` error that occurs when you try to create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to create an AKS cluster, you receive the following error message:

Reconcile VNet failed.

Details: VNetReconciler retry failed:

Category: ClientError; SubCode: LinkedAuthorizationFailed;

Dependency: Microsoft.Network/virtualNetworks; OrginalError:

Code="LinkedAuthorizationFailed"

Message="The client '12345678-1234-1234-1234-123456789098' with object id '123456789-1234-1234-1234-1234567890987' has permission to perform action 'Microsoft.Network/virtualNetworks/write' on scope '/subscriptions/<subscription-id-guid>/resourceGroups/MC_MyRG_westeurope/providers/Microsoft.Network/virtualNetworks/aks-vnet'; however, it does not have permission to perform action 'Microsoft.Network/ddosProtectionPlans/join/action' on the linked scope(s) '/subscriptions/<subscription-id-guid>/resourcegroups/ddos-protection-plan-rg/providers/microsoft.network/ddosprotectionplans/upmddosprotectionplan' or the linked scope(s) are invalid.";

AKSTeam: Networking, Retriable: false.

Cause

A service principal doesn't have permission to use a resource that's required for cluster creation.

Solution

Grant the service principal permissions to use the resource that's mentioned in the error message. The example output in the "Symptoms" section provides the following information.

Item	Value
Service principal	12345678-1234-1234-1234-123456789098
Resource	/subscriptions/<subscription-id-guid>/resourcegroups/ddos-protection-plan-rg/providers/microsoft.network/ddosprotectionplans/upmddosprotectionplan
Operation	Microsoft.Network/ddosProtectionPlans/join/action

For more information about how to grant permissions to the service principal, see [Assign Azure roles using the Azure portal](#).

More information

- [General troubleshooting of AKS cluster creation issues](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Troubleshoot LoadBalancerInUseByVirtualMachineScaleSet or NetworkSecurityGroupInUseByVirtualMachineScaleSet error code

Article • 05/19/2025

This article discusses how to identify and resolve the

`LoadBalancerInUseByVirtualMachineScaleSet` or

`NetworkSecurityGroupInUseByVirtualMachineScaleSet` error that occurs when you try to delete a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to delete an AKS cluster, you receive the following error message for

`LoadBalancerInUseByVirtualMachineScaleSet`:

```
internalErrorCode: "LoadBalancerInUseByVirtualMachineScaleSet"
```

```
StatusCode=409
```

```
{
```

```
"Cannot delete load balancer .../Microsoft.Network/loadBalancers/kubernetes since its child  
resources aksOutboundBackendPool, kubernetes are in use by virtual machine scale set  
.../Microsoft.Compute/virtualMachineScaleSets/aks-worker-test-vmss"
```

```
}
```

Or, you receive the following error message for

`NetworkSecurityGroupInUseByVirtualMachineScaleSet`:

```
internalErrorCode: "NetworkSecurityGroupInUseByVirtualMachineScaleSet"
```

```
StatusCode=409
```

```
{
```

```
"Cannot delete network security group .../Microsoft.Network/networkSecurityGroups/aks-  
agentpool since it is in use by virtual machine scale set"
```

```
.../Microsoft.Compute/virtualMachineScaleSets/aks-vmss"
```

```
}
```

Cause

You tried to delete an AKS cluster while the virtual machine scale set was still using the associated public IP address or network security group (NSG).

Solution

To fix this issue, use one of the following methods:

- Remove all public IP addresses that are associated with Azure Load Balancer. For more information, see [View, modify settings for, or delete a public IP address](#).
- Dissociate the NSG that's used by the subnet. For more information, see [Associate or dissociate a network security group](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the MissingSubscriptionRegistration error code

Article • 04/16/2025

This article discusses how to identify and resolve the `MissingSubscriptionRegistration` error that occurs when you try to create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to deploy an AKS cluster, you receive the following error message:

```
Code="MissingSubscriptionRegistration"
```

```
Message="The subscription is not registered to use namespace  
'Microsoft.OperationsManagement'. See https://aka.ms/rps-not-found for how to  
register subscriptions."
```

```
Details=[{
```

```
    "code": "MissingSubscriptionRegistration",  
  
    "message": "The subscription is not registered to use namespace  
'Microsoft.OperationsManagement'. See https://aka.ms/rps-not-found for how to  
register subscriptions.",  
  
    "target": "Microsoft.OperationsManagement"
```

```
}]
```

Cause

You receive this error message for one of the following reasons:

- The required resource provider isn't registered for your subscription.
- The API version isn't supported for the resource type.
- The location isn't supported for the resource type.

Solution

To resolve this issue, follow the instructions in the "Solution" section of [Resolve errors for resource provider registration](#). Replace the existing namespace with the namespace that's shown in the error message. In the example in the "Symptoms" section, the namespace is `Microsoft.OperationsManagement`.

More information

- [General troubleshooting of AKS cluster creation issues](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Can't upgrade AKS cluster because of the NodePoolMcVersionIncompatible error

Article • 05/19/2025

This article discusses how to resolve the "NodePoolMcVersionIncompatible - Node pool version 1.x.y and control plane version 1.a.b are incompatible" error that occurs when you upgrade a node pool in a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- [Azure CLI](#)

Symptoms

When you upgrade a node pool in an AKS cluster, you receive one of the following error messages:

BadRequest - NodePoolMcVersionIncompatible

Error: Node pool version 1.x.y and control plane version 1.a.b are incompatible. Minor version of node pool cannot be more than 2 versions less than control plane's version. Minor version of node pool is x and control plane is a. For more information, please check <https://aka.ms/version-skew-policy>.

Resource ID:

/subscriptions/<subscription_id>/resourcegroups/<aks_cluster_resource_group>/providers/Microsoft.ContainerService/managedClusters/<aks_cluster_name>.

BadRequest - NodePoolMcVersionIncompatible

Error: Node pool version 1.x.y and control plane version 1.a.b are incompatible. Minor version of node pool version x is bigger than control plane version a. For more information, please check <https://aka.ms/version-skew-policy>.

Resource ID:

/subscriptions/<subscription_id>/resourcegroups/<aks_cluster_resource_group>/providers/Microsoft.ContainerService/managedClusters/<aks_cluster_name>.

Cause

These issues occur if you try to upgrade a node pool that's more than two versions behind the AKS control plane version, or if you try to add a node pool that's at a more recent version than the control plane version.

You must meet the following conditions when you upgrade a node pool:

- The node pool version can't be greater than the control <*major*>.<*minor*>.<*patch*> version.
- The node pool version must be within two *minor* versions of the control plane version.

For more information, see the [AKS validation rules for upgrade](#).

Solution 1: Make sure that the node pool version is within two minor versions of the control plane version

1. Get the control plane version by running the `az aks get-upgrades` command in Azure CLI.

Here's an example use of the command. The `MasterVersion` output column contains the control plane version.

Output			
Name	ResourceGroup	MasterVersion	Upgrades
default	aksrg	1.23.12	1.23.15, 1.24.6, 1.24.9

2. Upgrade the node pool by running the `az aks nodepool upgrade` Azure CLI command, and provide a Kubernetes version that's within two minor versions of the control plane version.

For example, if the control plane version is `1.23.12`, you can specify the Kubernetes version of the node pool as `1.23.8` or `1.23.12`.

Here's an example use of the command:

```
Azure CLI
```

```
az aks nodepool upgrade \
--resource-group aksrg \
--cluster-name testcluster1 \
--name mynodepool \
--kubernetes-version 1.23.8 \
--no-wait
```

Solution 2: Make sure that the node pool version isn't greater than the control plane version

1. Get the control plane version by running the `az aks get-upgrades` command in Azure CLI.

Here's an example use of the command. The `MasterVersion` output column contains the control plane version.

Azure CLI

```
az aks get-upgrades --resource-group aksrg --name testcluster1 --output table
```

Output

Name	ResourceGroup	MasterVersion	Upgrades
default	aksrg	1.23.12	1.23.15, 1.24.6, 1.24.9

2. Upgrade the node pool by running the `az aks nodepool upgrade` Azure CLI command, and provide a Kubernetes version that's less than or equal to the control plane version.

Here's an example use of the command:

Azure CLI

```
az aks nodepool upgrade \
--resource-group aksrg \
--cluster-name testcluster1 \
--name mynodepool \
--kubernetes-version 1.23.12 \
--no-wait
```

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the OperationNotAllowed or PublicIPCountLimitReached quota error

Article • 04/16/2025

This article describes how to identify and resolve the `OperationNotAllowed` or `PublicIPCountLimitReached` quota error that occurs when you try to create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to create an AKS cluster, you receive the following error message:

`Code="OperationNotAllowed"`

`Message="Operation could not be completed as it results in exceeding approved standardEv3Family Cores quota."`

Additional details - Deployment Model: Resource Manager,

Location: uksouth,

Current Limit: 200,

Current Usage: 200,

Additional Required: 8,

(Minimum) New Limit Required: 208.

Submit a request for Quota increase at

`https://aka.ms/ProdportalCRP/#blade/Microsoft_Azure_Capacity/UsageAndQuota.ReactView/Parameters/<parameter-id>` by specifying parameters listed in the 'Details' section for deployment to succeed. Please read more about quota limits at
`https://learn.microsoft.com/azure/azure-supportability/per-vm-quota-requests"`

Or, you receive the following error message:

Reconcile standard load balancer failed.

Details: outboundReconciler retry failed: Category: ClientError; **SubCode: PublicIPCountLimitReached;**

Dependency: Microsoft.Network/PublicIPAddresses;

OrginalError: Code="PublicIPCountLimitReached"

Message="Cannot create more than 1000 public IP addresses for this subscription in this region." Details=[]; AKSTeam: Networking, Retriable: false.

Cause

You've exhausted the quota for a resource in an SKU's region.

Solution 1: Request a quota increase for your SKU

Typically, these error messages provide a link to create a support ticket to increase the quota for that SKU. In most cases, these requests are approved automatically, and the increased quota will be available in a short time.

If you don't have the link to request a quota increase, make the request in the Azure portal, as follows:

1. In the [Azure portal](#), search for and select **Quotas**.
2. Select **Microsoft.Compute** to view the list of quota records.
3. Use the **Location** list to filter for records from only your region.
4. In the specific record that's out-of-quota, select the pencil icon at the end of the record row.
5. In the **Request quota increase** pane, complete the **New limit** field, and then select **Submit**.

After your quota increase request is approved, you can retry the cluster creation operation.

Solution 2: Reprovision the cluster in another region or SKU

Alternatively, you can reprovision the cluster in a different region or SKU that has enough capacity for your cluster.

More information

- [General troubleshooting of AKS cluster creation issues](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the PublicIPCountLimitReached error code

Article • 05/19/2025

This article explains how to identify and resolve the "PublicIPCountLimitReached" error. This error occurs when you try to create, update, or upgrade an Azure Kubernetes Service (AKS) cluster. This error can also be caused by any other action that triggers the creation of a Public IP address, such as deploying a Kubernetes Service-type Public Load Balancer.

Symptoms

An AKS cluster creation, update, upgrade, or other operation that triggers the creation of a Public IP address, such as deploying a Kubernetes Service that has a Public Load Balancer, fails and returns a "PublicIPCountLimitReached" error message.

Cause

This error occurs if you've reached the maximum number of public IP addresses that are allowed for your subscription. The limit varies based on your subscription type. For more information about public IP address limits, refer to [this detailed guide](#).

Solution

To increase the public IP limit or quota for your subscription, follow these steps:

1. Navigate to the [Azure portal](#), and select the subscriptions for which you're performing the operation.
2. In the **Settings** section, select **Usage + quotas**. Set the **Provider** to **Networking**, and optionally filter by the region of your AKS cluster.
3. Locate the **Public IP Addresses** record. On the same line, select the **Create a new support request** button.

4. On the **New support request** page, specify the new limit that you require, and then follow the instructions to create the support request.

After the quota change takes effect, retry the operation that initially triggered the "PublicIPCountLimitReached" error.

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

A public IP address/subnet/network security group in use cannot be deleted

Article • 05/19/2025

This article discusses how to identify and resolve the `PublicIPAddressCannotBeDeleted`, `InUseSubnetCannotBeDeleted`, or `InUseNetworkSecurityGroupCannotBeDeleted` error that occurs when you try to delete a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to delete an AKS cluster, you receive one of the following error messages:

- For the `PublicIPAddressCannotBeDeleted` error code:

```
{
```

message: "Public IP address/providers/Microsoft.Network/publicIPAddresses/ can not be deleted since it is still allocated to resource/providers/Microsoft.Network/loadBalancers/kubernetes/frontendIPConfigurations /..... . In order to delete the public IP, disassociate/detach the Public IP address from the resource."

```
}
```

- For the `InUseSubnetCannotBeDeleted` error code:

```
{
```

message: "Subnet aks-subnet is in use by/Microsoft.Network/networkInterfaces/|providers|Microsoft.Compute|virtualMachines|eScaleSets|vmss|virtualMachines|1|networkInterfaces|aks-worker-vmss/ipConfigurations/ipconfig1 and cannot be deleted. In order to delete the subnet, delete all the resources within the subnet."

```
}
```

or

```
{
```

message: "Subnet aks-subnet is in use by/resourceGroups/.../providers/Microsoft.Network/virtualNetworks/.../subnets/.../serv

iceAssociationLinks/AppServiceLink and cannot be deleted. In order to delete the subnet, delete all the resources within the subnet. See aka.ms/deletesubnet."

}

- For the `InUseNetworkSecurityGroupCannotBeDeleted` error code:

```
{
```

message: "Network security group-/Microsoft.Network/networkSecurityGroups/test cannot be deleted because it is in use by the following resources:-/Microsoft.Network/virtualNetworks/test/subnets/test. In order to delete the Network security group, remove the association with the resource(s)."

```
}
```

Cause

The AKS cluster is associated with a subnet, network security group (NSG), or specific public IP address that's currently being used. This association prevents you from deleting the cluster.

Solution

- Remove all public IP addresses that are associated with Azure Load Balancer and the resource that's used by the subnet. For more information, see [View, modify settings for, or delete a public IP address](#).
- In the load balancer, remove the rules for **Load Balance rules**, **Health probes**, and **Backend pools**.
- For the NSG and subnet, remove all associated rules. For more information, see [Associate or dissociate a network security group to or from a subnet or network interface](#).
- If you're using an App Service plan with a subnet connected to the AKS cluster's VNET, you have to remove the associated App Service plan and its internal resources (such as Function App and SQL Azure database) and then retry deleting the AKS cluster.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the "Quotaexceeded" error code

Article • 04/16/2025

This article discusses how to identify and resolve the "QuotaExceeded" error that occurs when you try to upgrade an Azure Kubernetes Service (AKS) cluster.

Here's an example of the error message:

Operation results in exceeding quota limits of Core. Maximum allowed: X, Current in use: X, Additional requested: X

Prerequisites

This article requires Azure CLI version 2.0.65 or a later version. To find the version number, run `az --version`. If you have to install or upgrade Azure CLI, see [How to install the Azure CLI](#).

For more detailed information about the upgrade process, see the "Upgrade an AKS cluster" section in [Upgrade an Azure Kubernetes Service \(AKS\) cluster](#).

Symptoms

An AKS cluster upgrade fails, and you receive a "QuotaExceeded" error message.

Cause

The issue occurs if the quota is reached. Your subscription doesn't have available resources that are required for upgrading.

Solution

To raise the limit or quota for your subscription, go to the [Azure portal](#) and file a [Service and subscription limits \(quotas\)](#) support ticket. In this case, you have to submit a support ticket to increase the quota for compute cores.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

RequestDisallowedByPolicy error when deploying an AKS cluster

Article • 04/16/2025

This article discusses how to identify and resolve the `RequestDisallowedByPolicy` error that occurs when you try to create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to deploy an AKS cluster, you receive the following error message:

Resource request failed due to RequestDisallowedByPolicy. Please see [https://aka.ms/aks-](https://aka.ms/aks-requestdisallowedbypolicy)

[requestdisallowedbypolicy](#) for more details. The detailed error message:

Code="RequestDisallowedByPolicy"

Message="Resource 'MC_clustername' was disallowed by policy."

Cause

For security or compliance, your subscription administrators might assign policies that limit how resources are deployed. For example, your subscription might have a policy that prevents you from creating public IP addresses, network security groups, user-defined routes, or route tables. The error message includes the specific reason why the cluster creation was blocked.

! Note

Only you can manage the policies in your environment. Microsoft can't disable or bypass those policies.

Solution

To fix this issue, follow these steps:

1. Find the policy that blocks the action. These policies are listed in the error message.

The name of a policy assignment or definition is the last segment of the `id` string that's shown in the error message.

```
# Example
Code: RequestDisallowedByPolicy
Message: Resource 'resourcegroup' was disallowed by policy. Policy
identifiers: '[{"policyAssignment":{"name":"Not allowed resource
types","id":"/subscriptions/00000000-0000-0000-
000000000000/providers/Microsoft.Authorization/policyAssignments/000000000000
000000000000"}, "policyDefinition":{"name":"Not allowed resource
types","id":"/subscriptions/00000000-0000-0000-0000-
000000000000/providers/Microsoft.Authorization/policyDefinitions/not-allowed-
resourcetypes","version":"1.0.0"}}]'.
```

2. If possible, update your deployment to comply with the policy restrictions, and then retry the deployment. Alternatively, if you have permission to update policy, [add an exemption](#) to the policy.

To get details about the policy that blocked your cluster deployment, see [RequestDisallowedByPolicy error with Azure resource policy](#).

 **Note**

After you fix the policy that blocks the AKS cluster creation, run the `az aks update -g MyResourceGroup -n MyManagedCluster` command to change the cluster from a failed state to a successful state. This change reconciles the cluster and retries the last failed operation. For more information about clusters in a failed state, see [Troubleshoot Azure Kubernetes Service clusters or nodes in a failed state](#).

More information

[General troubleshooting of AKS cluster creation issues](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the ServicePrincipalValidationClientError error code

Article • 04/16/2025

This article discusses how to identify and resolve the `ServicePrincipalValidationClientError` error that might occur if you try to create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- [Azure CLI](#), version 2.0.59 or a later version. If Azure CLI is already installed, you can find the version number by running `az --version`.

Symptoms

When you try to deploy an AKS cluster, you receive the following error message:

adal: Refresh request failed. Status Code = '401'.

Response body: {

```
"error": "invalid_client",

"error_description": "AADSTS7000215: Invalid client secret provided. Ensure the secret
being sent in the request is the client secret value, not the client secret ID, for a secret
added to app '123456789-1234-1234-1234-1234567890987'.\r\n
```

Trace ID: 12345\r\n

Correlation ID: 6789\r\n

Timestamp: 2022-02-03 03:07:11Z",

"error_codes": [7000215],

"timestamp": "2022-02-03 03:07:11Z",

"trace_id": "12345",

"correlation_id": "6789",

"error_uri": "<https://login.microsoftonline.com/error?code=7000215>"

} Endpoint <https://login.microsoftonline.com/123456787/oauth2/token?api-version=1.0>

Cause

The secret that's provided for the highlighted service principal isn't valid.

Solution 1: Reset the service principal secret

To resolve this issue, reset the service principal secret by using one of the following methods:

- Reset the service principal's credential by running the [az ad sp credential reset](#) command:

Azure CLI

```
az ad sp credential reset --name "01234567-89ab-cdef-0123-456789abcdef" --query password --output tsv
```

- Specify the expiration date by running the following command:

Azure CLI

```
az ad sp credential reset --name <service-principal-name> --credential-description "New secret for AKS" --years 1
```

The preceding command resets the secret and displays it as output. Then, you can specify the new secret when you try to create the new cluster again.

For failed operations in an existing cluster, ensure that you update your AKS cluster with the new secret:

Azure CLI

```
az aks update-credentials --resource-group <resource-group> --name <aks-cluster> --reset-service-principal --client-secret <new-client-secret>
```

Solution 2: Create a new service principal

You can create a new service principal and get the secret that's associated with it by running the [az ad sp create-for-rbac](#) command:

Azure CLI

```
az ad sp create-for-rbac --role Contributor
```

The output of the command should resemble the following JSON string:

JSON

```
{  
  "appId": "12345678-9abc-def0-1234-56789abcdef0",  
  "name": "23456789-abcd-ef01-2345-6789abcdef01",  
  "password": "3456789a-bcde-f012-3456-789abcdef012",  
  "tenant": "456789ab-cdef-0123-4567-89abcdef0123"  
}
```

Note the `appId` and `password` values that are generated. After you get these values, you can rerun the cluster creation command for the new service principal and secret.

To update your AKS cluster with the new service principal's credential, run the following command:

Azure CLI

```
az aks update-credentials --resource-group <resource-group> --name <aks-cluster> -  
--service-principal <new-client-id> --client-secret <new-client-secret>
```

More information

- [General troubleshooting of AKS cluster creation issues](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the SubscriptionRequestsThrottled error code (429)

Article • 04/16/2025

This article discusses how to identify and resolve the `SubscriptionRequestsThrottled` error (status 429) that occurs when you try to create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to create an AKS cluster, you receive the following "Reconcile standard load balancer failed" error message that shows a "SubCode" value of `SubscriptionRequestsThrottled` and a "Status" value of **429**:

Reconcile standard load balancer failed.

Details: outboundReconciler retry failed:

Category: ClientError;

SubCode: `SubscriptionRequestsThrottled`;

Dependency: Microsoft.Network/PublicIPAddresses;

OrginalError: autorest/azure: Service returned an error. **Status=429**

Code="SubscriptionRequestsThrottled"

Message="Number of requests for subscription '*<subscription-id-guid>*' and operation 'GET/SUBSCRIPTIONS/RESOURCEGROUPS/PROVIDERS/MICROSOFT.NETWORK/PUBLICIPADDRESSES' exceeded the backend storage limit. Please try again after '6' seconds.";

AKSTeam: Networking, Retriable: false.

Request throttling can occur on different Azure components, so the error message might be different based on the kind of resource this issue is occurring on.

Cause

Azure Resource Manager requests are being throttled. For information about how Azure Resource Manager limits work, and the specific limits per hour, see [Throttling Resource Manager requests](#).

Solution 1: Use another subscription

If you have access to a different subscription, you can simply deploy the cluster to that subscription.

Solution 2: Modify your access patterns

To resolve this issue, examine your access patterns for the throttled subscription. The following table lists the possible access patterns and corresponding solutions.

 [Expand table](#)

Access pattern	Solution
Automated scripts constantly scan the subscription	Run the scripts less frequently
Many users access the subscription	Have each user use their own subscription
Scripts scan every storage account in the subscription	Scope the script to query only the resources that it must have

More information

- [General troubleshooting of AKS cluster creation issues](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the SubnetWithExternalResourcesCannotBeUsedByOtherResources error code

Article • 04/16/2025

This article provides solutions to the "SubnetWithExternalResourcesCannotBeUsedByOtherResources" error code that occurs when you create, update, or scale a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When attempting to create an AKS cluster or perform an "update" or "scale" operation on an existing AKS cluster, you may encounter the following error message:

Code: SubnetWithExternalResourcesCannotBeUsedByOtherResources
Error Message: Subnet <URI of subnet1> referenced by <URI of the AKS resource referencing the subnet1> cannot be used because it contains external resources. The external resources in this subnet are <URI of the external reference used by subnet1>. You need to delete these external resources before deploying into this subnet.

Cause

This error is related to the configuration of subnets. It means that you're trying to use a subnet that already has external resources associated with it for another resource. However, a subnet that has external resources associated with it isn't allowed to be used by other resources. For more information, see [Effect of subnet delegation on your subnet](#).

Solution

If you need to use the subnet for another resource, use one of the following methods:

- Create a new subnet and associate it with the new resource.
- Delete the external references, including service association links, subnet delegations, and so on.

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the ServiceCidrOverlapExistingSubnetsCidr error during an AKS cluster upgrade

Article • 05/19/2025

This article discusses how to identify and resolve the "ServiceCidrOverlapExistingSubnetsCidr" error that might occur when you try to [upgrade a Microsoft Azure Kubernetes Service \(AKS\) cluster](#).

Symptoms

An AKS cluster upgrade operation fails and displays the following error message:

```
(ServiceCidrOverlapExistingSubnetsCidr) The specified service CIDR <service-cidr-1> is  
conflicted with an existing subnet CIDR <subnet-cidr-2>  
Code: ServiceCidrOverlapExistingSubnetsCidr  
Message: The specified service CIDR <service-cidr-1> is conflicted with an existing subnet  
CIDR <subnet-cidr-2>  
Target: networkProfile.serviceCIDR
```

Cause

The service address range of a cluster is the set of virtual IP addresses that Kubernetes assigns to internal services in the cluster. This range is defined upon initial cluster creation. The range shouldn't overlap with the cluster's virtual network or any other network that can be routed from the cluster. For more information, see [Deployment parameters](#).

Before AKS starts an upgrade operation, it checks the cluster's virtual network for any existing subnet Classless Inter-Domain Routing (CIDR) address spaces that overlap with the cluster's service CIDR. If any such subnet overlap is found, the operation generates the "ServiceCidrOverlapExistingSubnetsCidr" error.

To resolve this issue, use one of the following solutions.

Solution 1: Remove the overlapping subnet

 Note

Use this solution if no resources are attached to the subnet.

1. Delete the subnet. To do this, follow the steps that are described in [Delete a subnet](#).
2. Retry the AKS cluster upgrade operation.

Solution 2: Adjust the overlapping subnet address range

(!) Note

Use this solution if it's acceptable to change the subnet's address range.

1. Change the subnet's address range. To do this, follow the steps that are described in [Change subnet settings](#).
2. Retry the AKS cluster upgrade operation.

Solution 3: Redeploy the cluster with a different service CIDR

(!) Note

Use this solution if it's not acceptable or not possible to remove the overlapping subnet or adjust its configuration. You can't change the cluster's service CIDR after cluster creation.

Review the [deployment parameters](#) and redeploy your cluster by using a different service CIDR.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the SubnetIsFull error code during an AKS cluster upgrade

Article • 05/19/2025

This article discusses how to identify and resolve the "SubnetIsFull" error that occurs when you try to upgrade an Azure Kubernetes Service (AKS) cluster.

Here's an example of the error message:

```
Failed to scale node pool <AGENT POOL NAME>' in Kubernetes service '<NAME>'. Error:  
VMSSAgentPoolReconciler retry failed: Code='SubnetIsFull' Message='<SUBNET NAME>\  
with address prefix <PREFIX>\ doesn't have enough capacity for IP addresses.' Details=[]
```

Prerequisites

This article requires Azure CLI version 2.0.65 or a later version. To find the version number, run `az --version`. If you have to install or upgrade Azure CLI, see [How to install the Azure CLI](#).

For more detailed information about the upgrade process, see the "Upgrade an AKS cluster" section in [Upgrade an Azure Kubernetes Service \(AKS\) cluster](#).

Symptoms

An AKS cluster upgrade fails, and you receive a "SubnetIsFull" error message.

Cause

This error occurs if your cluster doesn't have enough IP addresses to create a new node.

When you plan to do an upgrade or scaling operation, consider the number of required IP addresses. If the IP address range that you configured in the cluster supports only a fixed number of nodes, the upgrade or scaling operation will fail. For more information, see [IP address planning for your Azure Kubernetes Service \(AKS\) clusters](#).

Solution

Reduce the cluster nodes to reserve IP addresses for the upgrade.

If scaling down isn't an option, and your virtual network CIDR has enough IP addresses, try to add a node pool that has a [unique subnet](#):

1. Add a new user node pool in the virtual network on a larger subnet.
2. Switch the original node pool to a system node pool type.
3. Scale up the user node pool.
4. Scale down the original node pool.

More information

- [InsufficientSubnetSize error code](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Troubleshoot TooManyRequestsReceived or SubscriptionRequestsThrottled error code

Article • 05/19/2025

This article discusses how to identify and resolve the `TooManyRequestsReceived` or `SubscriptionRequestsThrottled` error that occurs when you try to delete a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to delete an AKS cluster, you receive the following error message:

```
internalErrorCode: TooManyRequestsReceived
```

```
StatusCode: 429
```

```
{
```

```
message: "Number of read requests for subscription '.....'" exceeded the limit of '....' for time  
interval 'XX:XX:XX'. Please try again after '.....' seconds."
```

```
}
```

Cause

Every subscription-level and tenant-level operation is subject to throttling limits. These limits apply to each instance of Azure Resource Manager. When you reach the limit, you receive an HTTP response that indicates status code 429: "Too many requests."

Solution

The HTTP response includes a `Retry-After` value. This specifies the number of seconds that your application should wait (or sleep) before it sends the next request. If you send a request before the retry value has elapsed, your request isn't processed, and a new retry value is returned. For more information about throttling limits, see [Throttling Resource Manager requests](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the Throttled error code (429)

Article • 04/16/2025

This article discusses how to identify and resolve the `Throttled` error (status 429) that occurs when you try to create and deploy a Microsoft Azure Kubernetes Service (AKS) cluster.

Symptoms

When you try to create an AKS cluster, you receive the following "The PutManagedClusterHandler.PUT request limit has been exceeded" error message that shows a "SubCode" value of **Throttled** and a "Status" value of **429**:

Category: ClientError;

SubCode: Throttled;

OrginalError: autorest/azure: Service returned an error. **Status=429**

Code="Throttled"

Message=> The PutManagedClusterHandler.PUT request limit has been exceeded for SubID='<*subscription-id-guid*>', please retry again in X seconds. For more information, please visit aka.ms/aks/throttling"; Request throttling can occur on various Azure components, so the error message might be different depending on the type of resource where this issue occurs.

Resource provider throttling is independent of ARM throttling and is tailored to the operations of a specific resource provider. In this scenario, AKS resource provider throttling is specific to the AKS resource provider and applies only to operations related to AKS resources.

Cause

AKS requests are throttled. For information about how AKS limits work and the specific limits per hour, see [Throttling limits on AKS resource provider APIs](#).

Solution

To resolve this issue, examine and modify your access pattern of the throttled subscription. The following table lists the possible access patterns and corresponding solutions.

Access pattern	Solution
Automated scripts constantly run LIST operations against managedCluster resources.	Run the scripts less frequently.
Users attempt to deploy multiple AKS clusters in a short period of time.	Space out deployments or use different subscriptions.
Users attempt to modify the same AKS cluster multiple times consecutively.	Space out operations. Ensure successful completion before initiating another one.
Users attempt to add, modify, or delete one or more agentPools on the same AKS cluster.	Space out operations. Ensure successful completion before initiating another one.

More information

[General troubleshooting of AKS cluster creation issues](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Error "UnsatisfiablePDB" when upgrading an AKS cluster

Article • 03/05/2025

This article discusses how to identify and resolve the "UnsatisfiablePDB" error that might occur when you try to [upgrade an Azure Kubernetes Service \(AKS\) cluster](#).

Prerequisites

This article requires Azure CLI version 2.53.0 or a later version. Run `az --version` to find your installed version. If you need to install or upgrade the Azure CLI, see [How to install the Azure CLI](#).

Symptoms

An AKS cluster upgrade operation fails with the following error message:

Code: UnsatisfiablePDB

Message: 1 error occurred:

* PDB < pdb-namespace >/< pdb-name > has maxunavailable == 0 can't proceed with put operation

Cause

Before starting an upgrade operation, AKS checks the cluster for any existing [Pod Disruption Budgets \(PDBs\)](#) that have the `maxUnavailable` parameter set to 0. Such PDBs are likely to block node drain operations. If node drain operations are blocked, the cluster upgrade operation can't complete successfully. This might potentially cause the cluster to be in a failed state.

After receiving the "UnsatisfiablePDB" error, you can confirm the PDB's status by running the following command:

Console

```
$ kubectl get pdb < pdb-name > -n < pdb-namespace >
```

The output of this command should be similar to the following one:

Output

NAME	MIN AVAILABLE	MAX UNAVAILABLE	ALLOWED DISRUPTIONS	AGE
<pdb-name>	N/A	0	0	49s

If the value of `MAX UNAVAILABLE` is 0, the node drain fails during the upgrade process.

To resolve this issue, use one of the following solutions.

Solution 1: Adjust the PDB's "maxUnavailable" parameter

ⓘ Note

Use this solution if you can edit the PDB resource directly.

1. Set the PDB's `maxUnavailable` parameter to `1` or a greater value. For more information, see [Specifying a PodDisruptionBudget](#).
2. Retry the AKS cluster upgrade operation.

Solution 2: Back up, delete, and redeploy the PDB

ⓘ Note

Use this solution if directly editing the PDB resource isn't viable.

1. Back up the PDB using the following command:

Console

```
$ kubectl get pdb < pdb-name > -n < pdb-namespace > -o yaml >  
pdb_backup.yaml
```

2. Delete the PDB using the following command:

Console

```
$ kubectl delete pdb < pdb-name > -n < pdb-namespace >
```

3. Retry the AKS cluster upgrade operation.
4. If the AKS cluster upgrade operation succeeds, redeploy the PDB using the following command:

Console

```
$ kubectl apply -f pdb_backup.yaml
```

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#)

Troubleshoot WINDOWS_CSE_ERROR_CHECK_API_SERVER_CONNECTIVITY error code (5)

Article • 04/16/2025

This article discusses how to identify and resolve the "WINDOWS_CSE_ERROR_CHECK_API_SERVER_CONNECTIVITY" error (5) that occurs when you try to add a Windows node pool in an Azure Kubernetes Service (AKS) cluster.

Prerequisites

One of the following tools is required:

- The PowerShell command-line shell for Windows nodes.
- The [Netcat](#) (nc) command-line tool for Linux nodes.

Symptoms

When you try to add a Windows node pool in an AKS cluster, you receive the following error message:

```
Code="VMExtensionProvisioningError"
Message="CSE Error: WINDOWS_CSE_ERROR_CHECK_API_SERVER_CONNECTIVITY." Exit
Code: 5. Details: Unable to establish connection from agents to Kubernetes API server.
```

Cause

Your cluster nodes can't connect to the cluster API server pod.

Troubleshooting steps

1. Verify that your nodes can resolve the cluster's fully qualified domain name (FQDN):

On existing Windows nodes, run the following command:

PowerShell

```
Test-NetConnection -ComputerName <cluster-fqdn> -Port 443
```

Or, on existing Linux nodes, run the following command:

Bash

```
nc -vz <cluster-fqdn> 443
```

2. If the command output shows `False` or `Timeout`, check your network configuration. For example, check whether you set "Deny" rules for the API server in network security groups (NSGs) of the virtual network.
3. If you're using egress filtering through a firewall, make sure that traffic is allowed to your cluster FQDN.
4. If you've authorized IP addresses that are enabled on your cluster, the firewall's outbound IP address can be blocked. In this scenario, you must add the outbound IP address of the firewall to the list of authorized IP ranges for the cluster. For more information, see [Secure access to the API server using authorized IP address ranges in AKS](#).

References

[General troubleshooting of AKS cluster creation issues](#)

Third-party information disclaimer

The third-party products that this article discusses are manufactured by companies that are independent of Microsoft. Microsoft makes no warranty, implied or otherwise, about the performance or reliability of these products.

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the ZonalAllocationFailed, AllocationFailed, or OverconstrainedAllocationRequest error code

Article • 04/17/2025

This article describes how to identify and resolve the `ZonalAllocationFailed`, `AllocationFailed`, or `OverconstrainedAllocationRequest` error that might occur when you try to create, deploy, or update a Microsoft Azure Kubernetes Service (AKS) cluster.

Prerequisites

- [Azure CLI](#) (optional), version 2.0.59 or a later version. If Azure CLI is already installed, you can find the version number by using `az --version`.
- [Azure PowerShell](#) (optional).

Symptoms

When you try to create an AKS cluster, you receive the following error message:

Reconcile vmss agent pool error: VMSSAgentPoolReconciler retry failed:

Category: InternalError;

SubCode: ZonalAllocationFailed;

Dependency: Microsoft.Compute/VirtualMachineScaleSet;

OrginalError: Code="ZonalAllocationFailed"

Message="Allocation failed. We do not have sufficient capacity for the requested VM size in this zone. Read more about improving likelihood of allocation success at <https://aka.ms/allocation-guidance>";

AKSTeam: NodeProvisioning

Or, when you try to upgrade or scale up a cluster, you receive the following error message:

Code="OverconstrainedAllocationRequest"

Message="Allocation failed. VM(s) with the following constraints cannot be allocated, because the condition is too restrictive. Please remove some constraints and try again."

Or, when you use dedicated hosts in a cluster and try to create or scale up a node pool, you receive the following error message:

Code="AllocationFailed"

Message="Allocation failed. VM allocation to the dedicated host failed. Please ensure that the dedicated host has enough capacity or try allocating elsewhere."

Cause 1: Limited zone availability in a SKU

You're trying to deploy, upgrade or scale up a cluster in a zone that has limited availability for the specific SKU.

Solution 1: Use a different SKU, zone, or region

Try one or more of the following methods:

- Redeploy the cluster in the same region by using a different SKU.
- Redeploy the cluster in a different zone in that region.
- Redeploy the cluster in a different region.
- Create a new node pool in a different zone or use a different SKU.

For more information about how to fix this error, see [Resolve errors for SKU not available](#).

Cause 2: Too many constraints for a virtual machine to accommodate

If you receive an `OverconstrainedAllocationRequest` error code, the Azure Compute platform can't allocate a new virtual machine (VM) to accommodate the required constraints. These constraints usually (but not always) include the following items:

- VM size
- VM SKU
- Accelerated networking
- Availability zone
- Ephemeral disk
- Proximity placement group (PPG)

Solution 2: Don't associate a proximity placement group with the node pool

If you receive an `OverconstrainedAllocationRequest` error code, you can try to create a new node pool that isn't associated with a proximity placement group.

Cause 3: Not enough dedicated hosts or fault domains

You're trying to deploy a node pool in a dedicated host group that has limited capacity or doesn't satisfy the fault domain constraint.

Solution 3: Ensure you have enough dedicated hosts for your AKS nodes/VMSS

As per [Planning for ADH Capacity on AKS](#), you're responsible for planning enough dedicated hosts to span as many fault domains as required by your AKS VMSS. For example, if the AKS VMSS is created with `FaultDomainCount=2`, you need at least two dedicated hosts in different fault domains (`FaultDomain 0` and `FaultDomain 1`).

More information

Ensuring capacity for users is a top priority for Microsoft, and we're working around the clock to reach this goal. The increasing popularity of Azure services emphasizes the need to scale up our infrastructure even more rapidly. With that in mind, we're expediting expansions and improving our resource deployment process to respond to strong customer demand. We're also adding a large amount of computing infrastructure monthly.

We have identified several methods to improve how we load-balance under a high-resource-usage situation and how to trigger the timely deployment of needed resources. Additionally, we're significantly increasing our capacity and will continue to plan for strong demand across all regions. For more information about the improvements that we're making toward delivering a resilient cloud supply chain, see [Advancing reliability through a resilient cloud supply chain](#).

References

- [General troubleshooting of AKS cluster creation issues](#)

- Virtual Machine Scale Sets - What to expect when using proximity placement groups
- Fix an AllocationFailed or ZonalAllocationFailed error when you create, restart, or resize Virtual Machine Scale Sets in Azure

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the SubnetIsDelegated error code

08/11/2025

This article discusses how to identify and resolve the SubnetIsDelegated error that occurs when you try to create a node pool.

Prerequisites

- Azure CLI (version 2.0.59 or a later version)

Symptoms

When you try to create a node pool in an AKS cluster, you receive the following error message:

Code: SubnetIsDelegated

Message: `AgentPoolProfile` subnet with id <subnet-id> cannot be used as it's a delegated subnet. Please check <https://aka.ms/adv-network-prerequest> for more details.

Cause

If you try to create a node pool by using a subnet, and the subnet is delegation-enabled for a particular Azure service, the new node pool can't be integrated with the AKS service.

Resolution

To resolve this issue, follow these steps:

1. Verify that the subnet is correctly delegated:

Bash

```
az network vnet subnet show \
--resource-group $RESOURCE_GROUP \
--vnet-name $VNET_NAME \
--name $SUBNET_NAME \
--query delegations
```

2. Make sure that the output shows `Microsoft.ContainerService/managedClusters` as the delegated service or no delegated service. If the output shows any other Azure service delegation, remove it by running the following command:

```
Bash
```

```
az network vnet subnet update \
--resource-group $RESOURCE_GROUP \
--vnet-name $VNET_NAME \
--name $SUBNET_NAME \
--remove delegations 0
```

3. Run the following command to add Managed Cluster delegation:

```
Bash
```

```
az network vnet subnet update \
--resource-group $RESOURCE_GROUP \
--vnet-name $VNET_NAME \
--name $SUBNET_NAME \
--delegations Microsoft.ContainerService/managedClusters
```

4. After the subnet delegation is removed, try again to create the node pool by using the `az aks nodepool add` command.

References

- [az aks node pool examples](#)

Contact us for help

If you have questions or need help, [create a support request](#), or ask Azure community support. You can also submit product feedback to [Azure feedback community](#).

Troubleshoot the VirtualNetworkNotInSucceededState error code

08/11/2025

Symptoms

When you create, upgrade, or scale an Azure Kubernetes Service (AKS) cluster or node pool, the deployment fails and returns an error message that resembles the following message:

Status=400 Code="VirtualNetworkNotInSucceededState"

Message="Set virtual network ownership failed. Subscription: <SUBSCRIPTION>; resource group: <RESOURCE GROUP>; virtual network name: <VNET NAME>. autorest/azure: Service returned an error. Status=400 Code="VirtualNetworkNotInSucceededState" Message="Virtual network /subscriptions/<SUBSCRIPTION>/resourceGroups/<RESOURCE GROUP>/providers/Microsoft.Network/virtualNetworks/<VNET> is in Updating state. It needs to be in Succeeded state in order to set resource ownership."

Cause

AKS can set ownership on a virtual network only if the `provisioningState` of the VNet is **Succeeded**. The request fails if the VNet is in the **Updating**, **Deleting**, or **Failed** state. Common causes for this condition include:

- Another create, update, or delete operation is still running on the VNet.
- A previous network operation failed and left the VNet in the **Failed** state.
- Multiple parallel cluster or node pool deployments are trying to modify the same VNet at the same time.

Resolution

Check the current provisioning state of the VNet:

.NET CLI

```
az network vnet show -g <resource-group> -n <vnet-name> --query
```

```
\\"provisioningState\\" -o tsv
```

If the command returns **Succeeded**, the VNet is fully set up and ready for use, and you can retry your AKS operation. If it returns any other value, the VNet might be in a failed or pending state that requires manual intervention. For more guidance, follow the troubleshooting steps in [Troubleshoot Azure Microsoft.Network failed provisioning state](#).

Contact us for help

If you have questions or need help, [create a support request](#), or ask [Azure community support](#). You can also submit product feedback to [Azure feedback community](#).