

Amazon ECS troubleshooting

You might need to troubleshoot issues with your load balancers, tasks, services, or container instances. This chapter helps you find diagnostic information from the Amazon ECS container agent, the Docker daemon on the container instance, and the service event log in the Amazon ECS console.

For information about stopped tasks, see one of the following.

Action	Learn more	
Resolve stopped task errors.	Viewing Amazon ECS stopped task errors	
View stopped task errors.	Resolve Amazon ECS stopped task errors	
Review stopped task error codes.	Amazon ECS stopped tasks error messages	
Review CannotPullContainer task errors.	CannotPullContainer task errors in Amazon ECS	
View task IAM role requests.	Viewing IAM role requests for Amazon ECS tasks	
Troubleshoot using task events.	Amazon ECS event capture in the console	

For information about service errors, see one of the following.

Action	Learn more	
View service event messages.	Viewing Amazon ECS service event messages	

Action	Learn more	
Review service event messages.	Amazon ECS service event messages	
Review load balancer issues.	Troubleshooting service load balancers in Amazon ECS	
Review service auto scaling issues.	Troubleshooting service auto scaling in Amazon ECS	

For information about task definition errors, see one of the following.

Action	Learn more	
Resolve task definition memory error.	Troubleshoot Amazon ECS task definition invalid CPU or memory errors	

For information about Amazon ECS agent errors, see one of the following.

Action	Learn more	
View Amazon ECS container agent logs.	Viewing Amazon ECS container agent logs	
Learn how to collect Amazon ECS logs.	Collecting container logs with Amazon ECS logs collector	
Retrieve diagnostic details with the Amazon ECS agent.	Retrieve Amazon ECS diagnostic details with agent introspection	

For information about Docker errors, see one of the following.

Action	Learn more	
Use Docker diagnostics.	Docker diagnostics in Amazon ECS	
Turn on Docker debug mode.	Configuring verbose output from the Docker daemon in Amazon ECS	
Troubleshoot Docker API error 500.	Troubleshoot the Docker API error (500): devmapper in Amazon ECS	

For information about ECS Exec and Amazon ECS Anywhere errors, see one of the following.

Action	Learn more	
Troubleshoot ECS Exec.	Troubleshoot Amazon ECS Exec issues	
Troubleshoot Amazon ECS Anywhere.	Troubleshoot Amazon ECS Anywhere issues	

For information about issues with attaching Amazon EBS volumes to Amazon ECS tasks, see the following:

Action	Learn more	
Troubleshooting Amazon EBS volume attachments to Amazon ECS tasks.	Troubleshooting Amazon EBS volume attachments to Amazon ECS tasks	
Status reasons for Amazon EBS volume attachments to Amazon ECS tasks.	Status reasons for Amazon EBS volume attachment to Amazon ECS tasks	

For information about issues with using shared AWS Cloud Map namespaces with Amazon ECS Service Connect, see one of the following:

Action	Learn more	
Troubleshooting Amazon ECS Service Connect with shared AWS Cloud Map namespaces.	Troubleshooting Amazon ECS Service Connect with shared AWS Cloud Map namespaces	

For information about throttling issues, see one of the following.

Action	Learn more	
Learn about Fargate throttling quotas.	AWS Fargate throttling quotas	
Learn the best practices for Amazon ECS throttling.	Handle Amazon ECS throttling issues	

For information about API errors, see one of the following.

Action	Learn more	
Resolve API errors.	Amazon ECS API failure reasons	

For information about AI-powered troubleshooting, see the following:

Action	Learn more	
Troubleshoot with Amazon Q Developer in the console.	Troubleshooting with Amazon Q Developer	

Action	Learn more	
Troubleshoot with AI assistants using the Amazon ECS MCP server.	Amazon ECS MCP server	

Resolve Amazon ECS stopped task errors

When your task fails to start, you see an error message in the console and in the describe-tasks output parameters (stoppedReason and stopCode).

You can view stopped tasks in the console for one hour. In order to see stopped tasks, you must change the filter option. For more information, see [Viewing Amazon ECS stopped task errors](#).

The following pages provide information about stopped tasks.

- Learn about changes to stopped task error messages.

[Amazon ECS stopped task error messages updates](#)

- View your stopped tasks so you can get information about the cause.

[Viewing Amazon ECS stopped task errors](#)

- Learn about the stopped tasks error messages and possible reasons for the errors.

[Amazon ECS stopped tasks error messages](#)

- Learn how to verify stopped task connectivity and fix the errors.

[Verifying Amazon ECS stopped task connectivity](#)

Amazon ECS stopped task error messages updates

Beginning June 14, 2024 the Amazon ECS team is changing the stopped task error messages as described in the following tables. The stopCode will not change. If your applications depend on exact error message strings, you must update your applications with the new strings. For help with questions or problems, contact AWS Support.

Note

We recommend that you do not rely on the error messages for your automation, because the error messages are subject to change.

CannotPullContainerError

Old error message	New error message	
CannotPullContainerError: Error response from daemon: pull access denied for <i>repository</i> , repository does not exist or may require 'docker login': denied: User: <i>roleARN</i>	<ul style="list-style-type: none">• CannotPullContainerError: The task can't pull the image. Check that the role has the permissions to pull images from the registry. Error response from daemon: pull access denied for <i>repository</i> , repository does not exist or may require 'docker login': denied: User: <i>roleARN</i> is not authorized to perform: ecr:BatchGetImage on resource: <i>image</i> because no identity-based policy allows the ecr:BatchGetImage action.• CannotPullContainerError: The task can't pull the image. Check whether the image exists. Error response from daemon: pull access denied for <i>repository</i> , repository does not exist or may require 'docker login':	

Old error message	New error message	
	denied: requested access to the resource is denied.	
CannotPullContainerError: Error response from daemon: Get <i>imageURI</i> : net/http: request canceled while waiting for connection	CannotPullContainerError: The task can't pull the image. Check your network configuration. Error response from daemon: Get <i>image</i> : net/http: request canceled while waiting for connection (Client.Timeout exceeded while awaiting headers)	
CannotPullContainerError: ref pull has been retried 5 time(s): failed to copy: httpReadSeeker: failed open: failed to do request: Get <i>registry-uri</i> : dial tcp <ip>:<port> i/o timeout	CannotPullContainerError: The task cannot pull <i>image-uri</i> from the registry <i>registry-uri</i> . There is a connection issue between the task and the registry. Check your task network configuration. : failed to copy: httpReadSeeker: failed open: failed to do request: Get <i>registry-uri</i> : dial tcp <ip>:<port> i/o timeout	

ResourceNotFoundException

Old error message	New error message	
Fetching secret data from AWS Secrets Manager in <i>region</i> region: secret <i>secretARN</i> : ResourceNotFoundException: Secrets	ResourceNotFoundException: The task can't retrieve the secret with ARN ' <i>secretARN</i> ' from AWS Secrets Manager. Check whether the	

Old error message	New error message	
Manager can't find the specified secret.	secret exists in the specified Region. ResourceNotFoundException: Fetching secret data from AWS Secrets Manager in region <i>region</i> : secret <i>secretARN</i> : ResourceNotFoundException: Secrets Manager can't find the specified secret.	

ResourceInitializationError

Old error message	New error message	
ResourceInitializationError : unable to pull secrets or registry auth: execution resource retrieval failed: unable to retrieve ecr registry auth: service call has been retried 3 time(s): RequestError: send request failed caused by: Post "https://api.ecr.us-east-1.amazonaws.com/": dial tcp <ip>:<port>: i/o timeout. Please check your task network configuration.	ResourceInitializationError : unable to pull secrets or registry auth: The task cannot pull registry auth from Amazon ECR: There is a connection issue between the task and Amazon ECR. Check your task network configuration. RequestError: send request failed caused by: Post "https://api.ecr.us-east-1.amazonaws.com/": dial tcp <ip>:<port>: i/o timeout	
ResourceInitializationError : unable to pull secrets or registry auth: execution resource retrieval failed: unable to retrieve secrets from ssm: service call has	ResourceInitializationError : unable to pull secrets or registry auth: unable to retrieve secrets from ssm: The task cannot pull secrets from AWS Systems Manager. There	

Old error message	New error message	
<p>been retried 5 time(s): RequestCanceled: request context canceled caused by: context deadline exceeded</p>	<p>is a connection issue between the task and AWS Systems Manager Parameter Store. Check your task network configuration. RequestCanceled: request context canceled caused by: context deadline exceeded</p>	
<p>ResourceInitializationError: failed to download env files: file download command: non empty error stream: RequestCanceled: request context canceled caused by: context deadline exceeded</p>	<p>ResourceInitializationError: failed to download env files: The task can't download the environment variable files from Amazon S3. There is a connection issue between the task and Amazon S3. Check your task network configuration. service call has been retried 5 time(s): RequestCanceled: request context canceled caused by: context deadline exceeded</p>	
<p>ResourceInitializationError : failed to validate logger args::signal:killed</p>	<p>ResourceInitializationError : failed to validate logger args: The task cannot find the Amazon CloudWatch log group defined in the task definition. There is a connection issue between the task and Amazon CloudWatch. Check your network configuration. : signal: killed</p>	

Old error message	New error message	
ResourceInitializationError : unable to pull secrets or registry auth: pull command failed: : signal: killed	ResourceInitializationError : unable to pull secrets or registry auth: Check your task network configuration. : signal: killed	

Viewing Amazon ECS stopped task errors

If you have trouble starting a task, your task might be stopping because of application or configuration errors. For example, you run the task and the task displays a PENDING status and then disappears.

If your task was created by an Amazon ECS service, the actions that Amazon ECS takes to maintain the service are published in the service events. You can view the events in the AWS Management Console, AWS CLI, AWS SDKs, the Amazon ECS API, or tools that use the SDKs and API. These events include Amazon ECS stopping and replaces a task because the containers in the task have stopped running, or have failed too many health checks from Elastic Load Balancing.

If your task ran on a container instance on Amazon EC2 or external computers, you can also look at the logs of the container runtime and the Amazon ECS Agent. These logs are on the host Amazon EC2 instance or external computer. For more information, see [Viewing Amazon ECS container agent logs](#).

Procedure

Console

AWS Management Console

The following steps can be used to check stopped tasks for errors using the console. In order to see stopped tasks, you must change the filter option.

Stopped tasks only appear in the console for 1 hour.

1. Open the console at <https://console.aws.amazon.com/ecs/v2>.
2. In the navigation pane, choose **Clusters**.

3. On the **Clusters** page, choose the cluster.
4. On the **Cluster : *name*** page, choose the **Tasks** tab.
5. Configure the filter to display stopped tasks. For **Filter desired status**, choose **Stopped**.

The **Stopped** option displays your stopped tasks and **Any desired status** displays all of your tasks.

6. Choose the stopped task to inspect.
7. In the row for your stopped task, in the **Last Status** column, choose **Stopped**.

A pop-up window displays the stopped reason.

AWS CLI

1. List the stopped tasks in a cluster. The output contains the Amazon Resource Name (ARN) of the task, which you need to describe the task.

```
aws ecs list-tasks \
  --cluster cluster_name \
  --desired-status STOPPED \
  --region region
```

2. Describe the stopped task to retrieve the information. For more information, see [describe-tasks](#) in the *AWS Command Line Interface Reference*.

```
aws ecs describe-tasks \
  --cluster cluster_name \
  --tasks arn:aws:ecs:region:account_id:task/cluster_name/task_ID \
  --region region
```

Use the following output parameters.

- **stopCode** - The stop code indicates why a task was stopped, for example `ResourceInitializationError`
- **StoppedReason** - The reason the task stopped.
- **reason** (in the `containers` structure) - The reason provide additional details about the stopped container.

Next steps

View your stopped tasks so you can get information about the cause. For more information, see [Amazon ECS stopped tasks error messages](#).

Amazon ECS stopped tasks error messages

The following are the possible error messages you may receive when your task stops unexpectedly.

To check your stopped tasks for an error message using the AWS Management Console, see [Viewing Amazon ECS stopped task errors](#).



Tip
You can use the [Amazon ECS MCP server](#) with AI assistants to analyze task failures and container logs using natural language.

Stopped task error codes have a category associated with them, for example "ResourceInitializationError". To get more information about each category, see the following:

Category	Learn more	
TaskFailedToStart	Troubleshooting Amazon ECS TaskFailedToStart errors	
ResourceInitializationError	Troubleshooting Amazon ECS ResourceInitializationError errors	
ResourceNotFoundException	Troubleshooting Amazon ECS ResourceNotFoundException errors	
SpotInterruptionError	Troubleshooting Amazon ECS SpotInterruption errors	
InternalError	Troubleshooting Amazon ECS InternalError errors	

Category	Learn more	
OutOfMemoryError	Troubleshooting Amazon ECS OutOfMemoryError errors	
ContainerRuntimeError	Troubleshooting Amazon ECS ContainerRuntimeError errors	
ContainerRuntimeTimeoutError	Troubleshooting Amazon ECS ContainerRuntimeTimeoutError errors	
CannotStartContainerError	Troubleshooting Amazon ECS CannotStartContainerError errors	
CannotStopContainerError	Troubleshooting Amazon ECS CannotStopContainerError errors	
CannotInspectContainerError	Troubleshooting Amazon ECS CannotInspectContainerError errors	
CannotCreateVolumeError	Troubleshooting Amazon ECS CannotCreateVolumeError errors	
CannotPullContainer	CannotPullContainer task errors in Amazon ECS	

Troubleshooting Amazon ECS TaskFailedToStart errors

The following are some TaskFailedToStart error messages and actions that you can take to fix the errors.

To check your stopped tasks for an error message using the AWS Management Console, see [Viewing Amazon ECS stopped task errors](#).

Unexpected EC2 error while attempting to Create Network Interface with public IP assignment enabled in subnet '*subnet-id*'

This happens when a Fargate task that uses the awsvpc network mode and runs in a subnet with a public IP address, and the subnet does not have enough IP addresses.

The number of available IP addresses is available on the subnet details page in the Amazon EC2 console, or by using [describe-subnets](#). For more information, see [View your subnet](#) in the *Amazon VPC User Guide*.

To fix this issue, you can create a new subnet to run your task in.

InternalError: *<reason>*

This error occurs when an ENI attachment is requested. Amazon EC2 asynchronously handles the provisioning of the ENI. The provisioning process takes time. Amazon ECS has a timeout in case there are long wait times or unreported failures. There are times when the ENI is provisioned, but the report comes to Amazon ECS after the failure timeout. In this case, Amazon ECS sees the reported task failure with an in-use ENI.

The selected task definition is not compatible with the selected compute strategy

This error occurs when you chose a task definition with a launch type that does not match the cluster capacity type. You need to select a task definition that matches the capacity provider assigned to your cluster.

Unable to attach network interface to unused device index

This error occurs when When using awsvpc networking type and there is not enough CPU/memory for the task. First, check the CPU for the instances. For more information, see [Amazon EC2 instance type specifications](#) in *Amazon EC2 instance types*. Take the CPU value for the instance and multiply it by the number of ENIs for the instance. Use that value e in the task definition.

AGENT

The container instance that you attempted to launch a task onto has an agent that's currently disconnected. To prevent extended wait times for task placement, the request was rejected.

For information about how to troubleshoot an agent that's disconnected, see [How do I troubleshoot a disconnected Amazon ECS agent](#).

Troubleshooting Amazon ECS ResourceInitializationError errors

The following are some ResourceInitialization error messages and actions that you can take to fix the errors.

To check your stopped tasks for an error message using the AWS Management Console, see [Viewing Amazon ECS stopped task errors](#).

Errors

- [The task cannot pull registry authentication from Amazon ECR. There is a connection issue between the task and Amazon ECR. Check your task network configuration.](#)
- [The task can't download the environment variable files from Amazon S3. There is a connection issue between the task and Amazon S3. Check your task network configuration.](#)
- [The task cannot pull secrets from AWS Systems Manager Parameter Store. Check your network connection between the task and AWS Systems Manager.](#)
- [The task can't pull secrets from AWS Secrets Manager. There is a connection issue between the task and Secrets Manager. Check your task network configuration.](#)
- [The task can't pull the secret from Secrets Manager. The task can't retrieve the secret with ARN 'secretARN' from Secrets Manager. Check whether the secret exists in the specified Region.](#)
- [pull command failed: unable to pull secrets or registry auth Check your task network configuration.](#)
- [The task cannot find the Amazon CloudWatch log group defined in the task definition. There is a connection issue between the task and Amazon CloudWatch. Check your network configuration.](#)
- [failed to initialize logging driver](#)
- [failed to invoke EFS utils commands to set up EFS volumes](#)

The task cannot pull registry authentication from Amazon ECR. There is a connection issue between the task and Amazon ECR. Check your task network configuration.

This error indicates that the task can't connect to Amazon ECR.

Check the connection between the task and Amazon ECR. For information, see [Verifying Amazon ECS stopped task connectivity](#).

The task can't download the environment variable files from Amazon S3. There is a connection issue between the task and Amazon S3. Check your task network configuration.

This error occurs when your task can't download your environment file from Amazon S3.

Check the connection between the task and the Amazon S3 VPC endpoint. For information, see [Verifying Amazon ECS stopped task connectivity](#).

The task cannot pull secrets from AWS Systems Manager Parameter Store. Check your network connection between the task and AWS Systems Manager.

This error occurs when your task can't pull the image defined in the task definition using the credentials in Systems Manager.

Check the connection between the task and the Systems Manager VPC endpoint. For information, see [Verifying Amazon ECS stopped task connectivity](#).

The task can't pull secrets from AWS Secrets Manager. There is a connection issue between the task and Secrets Manager. Check your task network configuration.

This error occurs when your task can't pull the image defined in the task definition using the credentials in Secrets Manager.

The error indicates that there is a network connectivity issue between the Systems Manager VPC endpoint and the task.

For information about how to verify the connectivity between the task and the endpoint, see [Verifying Amazon ECS stopped task connectivity](#).

The task can't pull the secret from Secrets Manager. The task can't retrieve the secret with ARN '*secretARN*' from Secrets Manager. Check whether the secret exists in the specified Region.

This error occurs when your task can't pull the image defined in the task definition using the credentials in Secrets Manager.

This issue is caused by one of the following reasons:

Error cause..	Do this...	
Network connectivity issue between the Secrets Manager VPC endpoint and the task.	Verify the connectivity between the task and the Secrets Manager endpoint.	

Error cause..	Do this...	
<p>The problem is a network issue when you see any of the following strings in the error message:</p> <ul style="list-style-type: none"> • dial tcp • dial udp • <ip>:<port>: i/o timeout • net/http: TLS handshake timeout • read: connection timed out • Client.Timeout exceeded while awaiting headers • net/http: request canceled while waiting for connection • signal: killed • context deadline exceeded 	<p>For more information, see Verifying Amazon ECS stopped task connectivity.</p>	
<p>The task execution role defined in the task definition doesn't have the permissions for Secrets Manager.</p>	<p>Add the required permissions for Secrets Manager to the task execution role. For more information, see Secrets Manager or Systems Manager permissions.</p>	
<p>The secret ARN doesn't exist</p>	<p>Check that the ARN exists in Secrets Manager. For information about viewing your images, see Find secrets in Secrets Manager in the <i>Secrets Manager Developer Guide</i>.</p>	

pull command failed: unable to pull secrets or registry auth Check your task network configuration.

This error occurs when your task can't connect to Amazon ECR, Systems Manager, or Secrets Manager. This is due to a misconfiguration in your network.

To fix this issue, verify the connectivity between the task and Amazon ECR. You also need to check connectivity between your task and the service which stores your secret (Systems Manager, or Secrets Manager). For more information, see [Verifying Amazon ECS stopped task connectivity](#).

The task cannot find the Amazon CloudWatch log group defined in the task definition. There is a connection issue between the task and Amazon CloudWatch. Check your network configuration.

This error occurs when your task fails to find the CloudWatch log group you defined in the task definition.

The error indicates that there is a network connectivity issue between the CloudWatch VPC endpoint and the task.

For information about how to verify the connectivity between the task and the endpoint, see [Verifying Amazon ECS stopped task connectivity](#).

failed to initialize logging driver

This error occurs when your task fails to find the CloudWatch log group you defined in the task definition.

The error indicates that the CloudWatch group in the task definition does not exist.

Use the following steps to find the missing CloudWatch.

1. Run the following command to get the task definition information.

```
aws ecs describe-task-definition \  
  --task-definition task-def-name
```

Look at the output for each container and note the `awslogs-group` value.

```
"logConfiguration": {  
    "logDriver": "awslogs",  
    "options": {
```

```
    "awslogs-group": "/ecs/example-group",
    "awslogs-create-group": "true",
    "awslogs-region": "us-east-1",
    "awslogs-stream-prefix": "ecs"
  },
```

2. Verify that the group exists in CloudWatch for more information, see [Working with log groups and log streams](#) in the *Amazon CloudWatch Logs User Guide*.

The issue is either that the group specified in the task definition is incorrect, or the log group does not exist.

3. Fix the issue.

The issue is...	Do this...	
The incorrect log group is specified in the task definition.	Update the task definition to include the log group configuration in the container definition.For information about updating the task definition, see Updating an Amazon ECS task definition using the console or RegisterTaskDefinition in the <i>Amazon Elastic Container Service API Reference</i> .	
The log group does not exist in CloudWatch	Create the log group. For more information, see Create a log group in CloudWatch Logs in the <i>Amazon CloudWatch Logs User Guide</i> .	

failed to invoke EFS utils commands to set up EFS volumes

The following issues might prevent you from mounting your Amazon EFS volumes on your asks:

- The Amazon EFS file system isn't configured correctly.
- The task doesn't have the required permissions.
- There are issues related to network and VPC configurations.

For information about how to debug and fix this issue, see [Why can't I mount my Amazon EFS volumes on my AWS Fargate tasks](#) on AWS re:Post.

Troubleshooting Amazon ECS ResourceNotFoundException errors

The following are some `ResourceNotFoundException` error messages and actions that you can take to fix the errors.

To check your stopped tasks for an error message using the AWS Management Console, see [Viewing Amazon ECS stopped task errors](#).

The task can't retrieve the secret with ARN '*secretARN*' from AWS Secrets Manager. Check whether the secret exists in the specified Region.

This error occurs when the task can't retrieve the secret from Secrets Manager. This means that the secret specified in the task definition (and contained in the error message) does not exist in Secrets Manager.

The Region is in the error message.

Fetching secret data from AWS Secrets Manager in region *region*: secret *secretARN*:
`ResourceNotFoundException: Secrets Manager can't find the specified secret.`

For information about finding a secret, see [Find secrets in AWS Secrets Manager](#) in the *AWS Secrets Manager User Guide*.

Use the following table to determine and address the error.

Issue	Actions	
The secret is in a different Region from the the task definition.	<ol style="list-style-type: none">Create the secret in the same Region as the task. For more information, see Create an AWS Secrets Manager secret.	

Issue	Actions	
	<p>b. Update the task definition with the new secret.</p> <p>For more information, see Updating an Amazon ECS task definition using the console or RegisterTaskDefinition in the <i>Amazon Elastic Container Service API Reference</i>.</p>	
<p>The task definition has the incorrect secret ARN. The correct secret exists in Secrets Manager.</p>	<p>Update the task definition with the correct secret.</p> <p>For more information, see Updating an Amazon ECS task definition using the console or RegisterTaskDefinition in the <i>Amazon Elastic Container Service API Reference</i>.</p>	
<p>The secret no longer exists.</p>	<p>a. Create the secret in the same Region as the task.</p> <p>For more information, see Create an AWS Secrets Manager secret.</p> <p>b. Update the task definition with the new secret.</p> <p>For more information, see Updating an Amazon ECS task definition using the console or RegisterTaskDefinition in the <i>Amazon Elastic Container Service API Reference</i>.</p>	

Troubleshooting Amazon ECS SpotInterruption errors

The SpotInterruption error has different reasons for Fargate and EC2s.

To check your stopped tasks for an error message using the AWS Management Console, see [Viewing Amazon ECS stopped task errors](#).

Fargate

The SpotInterruption error occurs when there is no Fargate Spot capacity or when Fargate takes back Spot capacity.

You can have your tasks run in multiple Availability Zones to allow for more capacity.

EC2

This error occurs when there are no available Spot Instances or EC2 takes back Spot Instance capacity.

You can have your instances run in multiple Availability Zones to allow for more capacity.

Troubleshooting Amazon ECS InternalError errors

Applies to: Fargate

To check your stopped tasks for an error message using the AWS Management Console, see [Viewing Amazon ECS stopped task errors](#).

The InternalError error when the agent encounters an unexpected, non-runtime related internal error.

This error only occurs if using platform version 1.4 or later.

For information about how to debug and fix this issue, see [Amazon ECS stopped tasks error messages](#).

Troubleshooting Amazon ECS OutOfMemoryError errors

The following are some OutOfMemoryError error messages and actions that you can take to fix the errors.

To check your stopped tasks for an error message using the AWS Management Console, see [Viewing Amazon ECS stopped task errors](#).

container killed due to memory usage

This error occurs when a container exits due to processes in the container consuming more memory than was allocated in the task definition, or due to host or operating system constraints.

Troubleshooting Amazon ECS ContainerRuntimeError errors

The following are some ContainerRuntimeError error messages and actions that you can take to fix the errors.

To check your stopped tasks for an error message using the AWS Management Console, see [Viewing Amazon ECS stopped task errors](#).

ContainerRuntimeError

This error occurs when the agent receives an unexpected error from `containerd` for a runtime-specific operation. This error is usually caused by an internal failure in the agent or the `containerd` runtime.

This error only occurs if you use platform version 1.4.0 or later (Linux) or 1.0.0 or later (Windows).

For information about how to debug and fix this issue, see [Why is my Amazon ECS task Stopped](#) on AWS re:Post.

Troubleshooting Amazon ECS ContainerRuntimeTimeoutError errors

The following are some ContainerRuntimeTimeoutError error messages and actions that you can take to fix the errors.

To check your stopped tasks for an error message using the AWS Management Console, see [Viewing Amazon ECS stopped task errors](#).

Could not transition to running; timed out after waiting 1m or Docker timeout error

This error occurs when a container can't transition to either a `RUNNING` or `STOPPED` state within the timeout period. The reason and timeout value is provided in the error message.

Troubleshooting Amazon ECS CannotStartContainerError errors

The following are some CannotStartContainerError error messages and actions that you can take to fix the errors.

To check your stopped tasks for an error message using the AWS Management Console, see [Viewing Amazon ECS stopped task errors](#).

failed to get container status: *<reason>*

This error occurs when a container can't be started.

If your container attempts to exceed the memory specified here, the container is stopped. Increase the memory presented to the container. This is the memory parameter in the task definition. For more information, see [the section called "Memory"](#).

Troubleshooting Amazon ECS CannotStopContainerError errors

The following are some CannotStopContainerError error messages and actions that you can take to fix the errors.

To check your stopped tasks for an error message using the AWS Management Console, see [Viewing Amazon ECS stopped task errors](#).

CannotStopContainerError

This error occurs when a container can't be stopped.

For information about how to debug and fix this issue, see [Why is my Amazon ECS task Stopped](#) on AWS re:Post.

Troubleshooting Amazon ECS CannotInspectContainerError errors

The following are some CannotInspectContainerError error messages and actions that you can take to fix the errors.

To check your stopped tasks for an error message using the AWS Management Console, see [Viewing Amazon ECS stopped task errors](#).

CannotInspectContainerError

This error occurs when the container agent can't describe the container through the container runtime.

When using platform version 1.3 or earlier, the Amazon ECS agent returns the reason from Docker.

When using platform version 1.4.0 or later (Linux) or 1.0.0 or later (Windows), the Fargate agent returns the reason from `containerd`.

For information about how to debug and fix this issue, see [Why is my Amazon ECS task Stopped](#) on AWS re:Post.

Troubleshooting Amazon ECS CannotCreateVolumeError errors

The following are some CannotCreateVolumeError error messages and actions that you can take to fix the errors.

To check your stopped tasks for an error message using the AWS Management Console, see [Viewing Amazon ECS stopped task errors](#).

CannotCreateVolumeError

This error occurs when the agent can't create the volume mount specified in the task definition.

This error only occurs if you use platform version 1.4.0 or later (Linux) or 1.0.0 or later (Windows).

For information about how to debug and fix this issue, see [Why is my Amazon ECS task Stopped](#) on AWS re:Post.

CannotPullContainer task errors in Amazon ECS

The following errors indicate that the task failed to start because Amazon ECS can't retrieve the specified container image.

Note

The 1.4 Fargate platform version truncates long error messages.

To check your stopped tasks for an error message using the AWS Management Console, see [Viewing Amazon ECS stopped task errors](#).

Tip

You can use the [Amazon ECS MCP server](#) with AI assistants to investigate image pull errors using natural language.

Errors

- [The task can't pull the image. Check that the role has the permissions to pull images from the registry.](#)
- [The task cannot pull 'image-name' from the Amazon ECR repository 'repository URI'. There is a connection issue between the task and Amazon ECR. Check your task network configuration.](#)
- [The task can't pull the image. Check your network configuration](#)
- [CannotPullContainerError: pull image manifest has been retried 5 time\(s\): failed to resolve ref](#)
- [API error \(500\): Get https://111122223333.dkr.ecr.us-east-1.amazonaws.com/v2/: net/http: request canceled while waiting for connection](#)
- [API error](#)
- [write /var/lib/docker/tmp/GetImageBlob111111111: no space left on device](#)
- [ERROR: toomanyrequests: Too Many Requests or You have reached your pull rate limit.](#)
- [Error response from daemon: Get url: net/http: request canceled while waiting for connection](#)
- [ref pull has been retried 1 time\(s\): failed to copy: httpReaderSeeker: failed open: unexpected status code](#)
- [pull access denied](#)
- [pull command failed: panic: runtime error: invalid memory address or nil pointer dereference](#)
- [error pulling image conf/error pulling image configuration](#)
- [Context canceled](#)

The task can't pull the image. Check that the role has the permissions to pull images from the registry.

This error indicates that the task can't pull the image specified in the task definition because of permission issues.

To resolve this issue:

1. Check that the image exists in the *repository*. For information about viewing your images, see [Viewing image details in Amazon ECR](#) in the *Amazon Elastic Container Registry User Guide*.
2. Verify that the *role-arn* has the correct permissions to pull the image.

For information about how to update roles, see [Update permissions for a role](#) in the *AWS Identity and Access Management Use Guide*.

The task uses one of the following roles:

- For tasks with the Fargate, this is the task execution role. For information about the additional permissions for Amazon ECR, [Fargate tasks pulling Amazon ECR images over interface endpoints permissions](#).
- For tasks with EC2, this is the container instance role. For information about the additional permissions for Amazon ECR, [Amazon ECR permissions](#).

The task cannot pull '*image-name*' from the Amazon ECR repository '*repository URI*'. There is a connection issue between the task and Amazon ECR. Check your task network configuration.

This error indicates that the task can't connect to Amazon ECR. Check the connection to the *repository URI* repository.

For information about how to verify and resolve the issue, see [Verifying Amazon ECS stopped task connectivity](#).

The task can't pull the image. Check your network configuration

This error indicates that the task can't connect to Amazon ECR.

For information about how to verify and resolve the issue, see [Verifying Amazon ECS stopped task connectivity](#).

CannotPullContainerError: pull image manifest has been retried 5 time(s): failed to resolve ref

This error indicates that the task can't pull the image.

To resolve this, you can:

- Verify that the image specified in the task definition matches the image in the repository.
- Amazon ECS forces image version stability. If the original image is no longer available you get this error. The image tag is part of enforcing this behavior. Change the image in the task definition from using :latest as the tag to a specific version. For more information, see [Container image resolution](#).

For information about how to verify and resolve the issue, see [Verifying Amazon ECS stopped task connectivity](#).

API error (500): Get https://111122223333.dkr.ecr.us-east-1.amazonaws.com/v2/: net/http: request canceled while waiting for connection

This error indicates that a connection timed out, because a route to the internet doesn't exist.

To resolve this issue, you can:

- For tasks in public subnets, specify **ENABLED** for **Auto-assign public IP** when launching the task. For more information, see [Running an application as an Amazon ECS task](#).
- For tasks in private subnets, specify **DISABLED** for **Auto-assign public IP** when launching the task, and configure a NAT gateway in your VPC to route requests to the internet. For more information, see [NAT Gateways](#) in the *Amazon VPC User Guide*.

API error

This error indicates that there is a connection issue with the Amazon ECR endpoint.

For information about how to resolve this issue, see [How can I resolve the Amazon ECR error "CannotPullContainerError: API error" in Amazon ECS](#) on the Support website.

write /var/lib/docker/tmp/**GetImageBlob111111111**: no space left on device

This error indicates that there is insufficient disk space.

To resolve this issue, free up disk space.

If you are using the Amazon ECS-optimized AMI, you can use the following command to retrieve the 20 largest files on your file system:

```
du -Sh / | sort -rh | head -20
```

Example output:

```
5.7G    /var/lib/docker/
containers/50501b5f4cbf90b406e0ca60bf4e6d4ec8f773a6c1d2b451ed8e0195418ad0d2
1.2G    /var/log/ecs
594M    /var/lib/docker/devicemapper/mnt/
c8e3010e36ce4c089bf286a623699f5233097ca126ebd5a700af023a5127633d/rootfs/data/logs
...
```

In some cases, the root volume might be filled out by a running container. If the container is using the default `json-file` log driver without a `max-size` limit, it may be that the log file is responsible for most of that space used. You can use the `docker ps` command to verify which container is using the space by mapping the directory name from the output above to the container ID. For example:

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS	NAMES	
50501b5f4cbf	amazon/amazon-ecs-agent:latest	"/agent"	4 days ago
Up 4 days		ecs-agent	

By default, when using the `json-file` log driver, Docker captures the standard output (and standard error) of all of your containers and writes them in files using the JSON format. You can set the `max-size` as a log driver option, which prevents the log file from taking up too much space. For more information, see [JSON File logging driver](#) in the Docker documentation.

The following is a container definition snippet showing how to use this option:

```
{
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "256m"
  }
}
```

An alternative, if your container logs are taking up too much disk space, is to use the `awslogs` log driver. The `awslogs` log driver sends the logs to CloudWatch, which frees up the disk space that would otherwise be used for your container logs on the container instance. For more information, see [Send Amazon ECS logs to CloudWatch](#).

You might need to update the disk size that Docker can access.

For more information, see [CannotPullContainerError: no space left on device](#).

ERROR: toomanyrequests: Too Many Requests or You have reached your pull rate limit.

This error indicates that there is a Docker Hub rate limiting.

If you receive one of the following errors, you're likely hitting the Docker Hub rate limits:

For more information about the Docker Hub rate limits, see [Understanding Docker Hub rate limiting](#).

If you have increased the Docker Hub rate limit and you need to authenticate your Docker pulls for your container instances, see [Private registry authentication for container instances](#).

Error response from daemon: Get *url*: net/http: request canceled while waiting for connection

This error indicates that a connection timed out, because a route to the internet doesn't exist.

To resolve this issue, you can:

- For tasks in public subnets, specify **ENABLED** for **Auto-assign public IP** when launching the task. For more information, see [Running an application as an Amazon ECS task](#).
- For tasks in private subnets, specify **DISABLED** for **Auto-assign public IP** when launching the task, and configure a NAT gateway in your VPC to route requests to the internet. For more information, see [NAT Gateways](#) in the *Amazon VPC User Guide*.

ref pull has been retried 1 time(s): failed to copy: httpReaderSeeker: failed open: unexpected status code

This error indicates that there was a failure when copying an image.

To resolve this issue, review one of the following articles:

- For Fargate tasks, see [How do I resolve the "cannotpullcontainererror" error for my Amazon ECS tasks on Fargate](#).
- For other tasks, see [How do I resolve the "cannotpullcontainererror" error for my Amazon ECS tasks](#).

pull access denied

This error indicates that there is no access to the image.

To resolve this issue, you might need to authenticate your Docker client with Amazon ECR. For more information, see [Private registry authentication](#) in the *Amazon ECR User Guide*.

pull command failed: panic: runtime error: invalid memory address or nil pointer dereference

This error indicates that there is no access to the image because of an invalid memory address or nil pointer dereference.

To resolve this issue:

- Check that you have the security group rules to reach Amazon S3.
- When you use gateway endpoints, you must add a route in the route table to access the endpoint.

error pulling image conf/error pulling image configuration

This error indicates a rate limit has been reached or there is a network error:

To resolve this issue, see [How can I resolve the "CannotPullContainerError" error in my Amazon ECS EC2 Launch Type Task.](#)

Context canceled

This error indicates that the context was cancelled.

The common cause for this error is because the VPC your task is using doesn't have a route to pull the container image from Amazon ECR.

Verifying Amazon ECS stopped task connectivity

There are times when a task stops because of a network connectivity issue. It might be an intermittent issue, but it is most likely caused because the task cannot connect to an endpoint.

Testing the task connectivity

You can use `AWSSupport-TroubleshootECSTaskFailedToStart` runbook to test the task connectivity. When you use the runbook, you need the following resource information:

- The task ID
Use the ID of the most recent failed task.
- The cluster that the task was in

For information about how to use the runbook, see [AWSSupport-TroubleshootECSTaskFailedToStart](#) in the *AWS Systems Manager Automation runbook reference*.

The runbook analyzes the task. You can view the results in the **Output** section for the following issues that can prevent a task from starting:

- Network connectivity to the configured container registry

- VPC endpoint connectivity
- Security group rule configuration

Fixing VPC endpoint issues

When the `AWSSupport-TroubleshootECSTaskFailedToStart` runbook result indicates the VPC endpoint issue, check the following configuration:

- The VPC where you create the endpoint and the VPC endpoint need to use Private DNS.
- Make sure that you have a AWS PrivateLink endpoint for the service that the task cannot connect to in the same VPC as the task. For more information see one of the following:

Service	VPC endpoint information for the service	
Amazon ECR	Amazon ECR interface VPC endpoints (AWS PrivateLink)	
Systems Manager	Improve the security of EC2 instances by using VPC endpoints for Systems Manager	
Secrets Manager	Using an AWS Secrets Manager VPC endpoint	
CloudWatch	CloudWatch VPC endpoint	
Amazon S3	AWS PrivateLink for Amazon S3	

- Configure an outbound rule for the task subnet which allows HTTPS on port 443 DNS (TCP) traffic. For more information, see [Configure security group rules](#) in the *Amazon Elastic Compute Cloud User Guide*.
- If you use a custom name domain server, then confirm the DNS query's settings. The query must have outbound access on port 53, and use UDP and TCP protocol. Also, it must have HTTPS access on port 443. For more information, see [Configure security group rules](#) in the *Amazon Elastic Compute Cloud User Guide*.

- If the subnet has a network ACL, the following ACL rules are required:
 - An outbound rule that allows traffic that allows traffic on ports 1024-65535.
 - An inbound rule that allows TCP traffic on port 443.

For information about how to configure rules, see [Control traffic to subnets using network ACLs](#) in the *Amazon Virtual Private Cloud User Guide*.

Fixing network issues

When the `AWSSupport-TroubleshootECSTaskFailedToStart` runbook result indicates a network issue, check the following configuration:

Tasks that use `awsvpc` network mode in a public subnet

Perform the following configuration based on the runbook:

- For tasks in public subnets, specify **ENABLED** for **Auto-assign public IP** when launching the task. For more information, see [Running an application as an Amazon ECS task](#).
- You need a gateway to handle internet traffic. The route table for the task subnet needs to have a route for traffic to the gateway.

For more information, see [Add and remove routes from a route table](#) in the *Amazon Virtual Private Cloud User Guide*.

Gateway type	Route table destination	Route table target
NAT	0.0.0.0/0	NAT gateway ID
Internet gateway	0.0.0.0/0	Internet gateway ID

- If the task subnet has a network ACL, the following ACL rules are required:
 - An outbound rule that allows traffic on ports 1024-65535.
 - An inbound rule that allows TCP traffic on port 443.

For information about how to configure rules, see [Control traffic to subnets using network ACLs](#) in the *Amazon Virtual Private Cloud User Guide*.

Tasks that use awsvpc network mode in a private subnet

Perform the following configuration based on the runbook:

- Choose **DISABLED** for **Auto-assign public IP** when launching the task.
- Configure a NAT gateway in your VPC to route requests to the internet. For more information, see [NAT Gateways](#) in the *Amazon Virtual Private Cloud User Guide*.
- The route table for the task subnet needs to have a route for traffic to the NAT gateway.

For more information, see [Add and remove routes from a route table](#) in the *Amazon Virtual Private Cloud User Guide*.

Gateway type	Route table destination	Route table target
NAT	0.0.0.0/0	NAT gateway ID

- If the task subnet has a network ACL, the following ACL rules are required:
 - An outbound rule that allows traffic on ports 1024-65535.
 - An inbound rule that allows TCP traffic on port 443.

For information about how to configure rules, see [Control traffic to subnets using network ACLs](#) in the *Amazon Virtual Private Cloud User Guide*.

Tasks that don't use awsvpc network mode in a public subnet

Perform the following configuration based on the runbook:

- Choose **Turn on** for **Auto assign IP** under **Networking for Amazon EC2 instances** when you create the cluster.

This option assigns a public IP address to the instance primary network interface.

- You need a gateway to handle internet traffic. The route table for the instance subnet needs to have a route for traffic to the gateway.

For more information, see [Add and remove routes from a route table](#) in the *Amazon Virtual Private Cloud User Guide*.

Gateway type	Route table destination	Route table target
NAT	0.0.0.0/0	NAT gateway ID
Internet gateway	0.0.0.0/0	Internet gateway ID

- If the instance subnet has a network ACL, the following ACL rules are required:
 - An outbound rule that allows traffic on ports 1024-65535.
 - An inbound rule that allows TCP traffic on port 443.

For information about how to configure rules, see [Control traffic to subnets using network ACLs](#) in the *Amazon Virtual Private Cloud User Guide*.

Tasks that don't use awsipc network mode in a private subnet

Perform the following configuration based on the runbook:

- Choose **Turn off** for **Auto assign IP** under **Networking for Amazon EC2 instances** when you create the cluster.
- Configure a NAT gateway in your VPC to route requests to the internet. For more information, see [NAT Gateways](#) in the *Amazon VPC User Guide*.
- The route table for the instance subnet needs to have a route for traffic to the NAT gateway.

For more information, see [Add and remove routes from a route table](#) in the *Amazon Virtual Private Cloud User Guide*.

Gateway type	Route table destination	Route table target
NAT	0.0.0.0/0	NAT gateway ID

- If the task subnet has a network ACL, the following ACL rules are required:
 - An outbound rule that allows traffic on ports 1024-65535.
 - An inbound rule that allows TCP traffic on port 443.

For information about how to configure rules, see [Control traffic to subnets using network ACLs](#) in the *Amazon Virtual Private Cloud User Guide*.

Viewing IAM role requests for Amazon ECS tasks

When you use a provider for your task credentials in an IAM role, the provider requests saved in an audit log. The audit log inherits the same log rotation settings as the container agent log. The `ECS_LOG_ROLLOVER_TYPE`, `ECS_LOG_MAX_FILE_SIZE_MB`, and `ECS_LOG_MAX_ROLL_COUNT` container agent configuration variables can be set to affect the behavior of the audit log. For more information, see [Amazon ECS container agent log configuration parameters](#).

For container agent version 1.36.0 and later, the audit log is located at `/var/log/ecs/audit.log`. When the log is rotated, a timestamp in `YYYY-MM-DD-HH` format is added to the end of the log file name.

For container agent version 1.35.0 and earlier, the audit log is located at `/var/log/ecs/audit.log.YYYY-MM-DD-HH`.

The log entry format is as follows:

- Timestamp
- HTTP response code
- IP address and port number of request origin
- Relative URI of the credential provider
- The user agent that made the request
- The ARN of the task to which the requesting container belongs
- The `GetCredentials` API name and version number
- The name of the Amazon ECS cluster to which the container instance is registered
- The container instance ARN

You can use the following command to view the log files.

```
cat /var/log/ecs/audit.log.2016-07-13-16
```

Output:

```
2016-07-13T16:11:53Z 200 172.17.0.5:52444 "/v1/credentials" "python-requests/2.7.0  
CPython/2.7.6 Linux/4.4.14-24.50.amzn1.x86_64" TASK_ARN GetCredentials  
1 CLUSTER_NAME CONTAINER_INSTANCE_ARN
```

Viewing Amazon ECS service event messages

When troubleshooting a problem with a service, the first place you should check for the diagnostic information is the service event log. You can view service events using the `DescribeServices` API, the AWS CLI, or by using the AWS Management Console.

When viewing service event messages using the Amazon ECS API, only the events from the service scheduler are returned. These include the most recent task placement and instance health events. However, the Amazon ECS console displays service events from the following sources.

- Task placement and instance health events from the Amazon ECS service scheduler. These events have a prefix of **service** (*service-name*). To ensure that this event view is helpful, we only show the 100 most recent events. Duplicate event messages are omitted until either the cause is resolved, or six hours passes. If the cause is not resolved within six hours, you receive another service event message for that cause.
- Service Auto Scaling events. These events have a prefix of **Message** and occur only when a service is configured with an Application Auto Scaling scaling policy.

Tip

You can use the [Amazon ECS MCP server](#) with AI assistants to analyze service events using natural language.

Use the following steps to view your current service event messages.

Console

1. Open the console at <https://console.aws.amazon.com/ecs/v2>.
2. In the navigation pane, choose **Clusters**.
3. On the **Clusters** page, choose the cluster.
4. Choose the service to inspect.
5. On the **Events** tab, view the messages.

AWS CLI

Use the [describe-services](#) command to view the service event messages for a specified service.

The following AWS CLI example describes the *service-name* service in the *default* cluster, which will provide the latest service event messages.

```
aws ecs describe-services \
  --cluster default \
  --services service-name \
  --region us-west-2
```

Amazon ECS service event messages

The following are examples of service event messages you may see in the Amazon ECS console.

service (*service-name*) has reached a steady state.

The service scheduler sends a service (*service-name*) has reached a steady state. service event when the service is healthy and at the desired number of tasks, thus reaching a steady state.

The service scheduler reports the status periodically, so you might receive this message multiple times.

service (*service-name*) was unable to place a task because no container instance met all of its requirements.

The service scheduler sends this event message when it couldn't find the available resources to add another task. The possible causes for this are:

Use capacity providers to automatically scale your EC2 instances. For more information, see [Amazon ECS capacity providers for EC2 workloads](#).

If you intended to use a capacity provider, make sure that you're either passing a capacity provider strategy or have a default capacity provider strategy associated with your cluster and are not passing launch type and capacity provider strategy as input

No container instances were found in your cluster

If no container instances are registered in the cluster you attempt to run a task in, you receive this error. You should add container instances to your cluster. For more information, see [Launching an Amazon ECS Linux container instance](#).

Not enough ports

If your task uses fixed host port mapping (for example, your task uses port 80 on the host for a web server), you must have at least one container instance per task, because only one container can use a single host port at a time. You should add container instances to your cluster or reduce your number of desired tasks.

Too many ports registered

The closest matching container instance for task placement can't exceed the maximum allowed reserved port limit of 100 host ports per container instance. Using dynamic host port mapping may remediate the issue.

Port already in-use

The task definition of this task uses the same port in its port mapping as a task already running on the container instance that was chosen. The service event message would have the chosen container instance ID as part of the message below.

```
The closest matching container-instance is already using a port required by your task.
```

Not enough memory

If your task definition specifies 1000 MiB of memory, and the container instances in your cluster each have 1024 MiB of memory, you can only run one copy of this task per container instance. You can experiment with less memory in your task definition so that you could launch more than one task per container instance, or launch more container instances into your cluster.

Note

If you're trying to maximize your resource utilization by providing your tasks as much memory as possible for a particular instance type, see [Reserving Amazon ECS Linux container instance memory](#).

Not enough CPU

A container instance has 1,024 CPU units for every CPU core. If your task definition specifies 1,000 CPU units, and the container instances in your cluster each have 1,024 CPU units, you can only run one copy of this task per container instance. You can experiment with fewer CPU units

in your task definition so that you could launch more than one task per container instance, or launch more container instances into your cluster.

Not enough available ENI attachment points

Tasks that use the `awsvpc` network mode each receive their own elastic network interface (ENI), which is attached to the container instance that hosts it. Amazon EC2 instances have a limit to the number of ENIs that can be attached to them and there are no container instances in the cluster that have ENI capacity available.

The ENI limit for individual container instances depends on the following conditions:

- If you **have not** opted in to the `awsvpcTrunking` account setting, the ENI limit for each container instance depends on the instance type. For more information, see [IP Addresses Per Network Interface Per Instance Type](#) in the *Amazon EC2 User Guide*.
- If you **have** opted in to the `awsvpcTrunking` account setting but you **have not** launched new container instances using a supported instance type after opting in, the ENI limit for each container instance is still at the default value. For more information, see [IP Addresses Per Network Interface Per Instance Type](#) in the *Amazon EC2 User Guide*.
- If you **have** opted in to the `awsvpcTrunking` account setting and you **have** launched new container instances using a supported instance type after opting in, additional ENIs are available. For more information, see [Supported instances for increased Amazon ECS container network interfaces](#).

For more information about opting in to the `awsvpcTrunking` account setting, see [Increasing Amazon ECS Linux container instance network interfaces](#).

You can add container instances to your cluster to provide more available network adapters.

Container instance missing required attribute

Some task definition parameters require a specific Docker remote API version to be installed on the container instance. Others, such as the logging driver options, require the container instances to register those log drivers with the `ECS_AVAILABLE_LOGGING_DRIVERS` agent configuration variable. If your task definition contains a parameter that requires a specific container instance attribute, and you don't have any available container instances that can satisfy this requirement, the task can't be placed.

A common cause of this error is if your service is using tasks that use the `awsvpc` network mode and EC2. The cluster you specified doesn't have a container instance registered to it in the same subnet that was specified in the `awsvpcConfiguration` when the service was created.

You can use the `AWSSupport-TroubleshootECSContainerInstance` runbook to troubleshoot. The runbook reviews whether the user data for the instance contains the correct cluster information, whether the instance profile contains the required permissions, and network configuration issues. For more information, see [AWSSupport-TroubleshootECSContainerInstance](#) in the *AWS Systems Manager Automation runbook reference User Guide*.

For more information on which attributes are required for specific task definition parameters and agent configuration variables, see [Amazon ECS task definition parameters for Fargate](#) and [Amazon ECS container agent configuration](#).

service (*service-name*) was unable to place a task because no container instance met all of its requirements. The closest matching container-instance *container-instance-id* has insufficient CPU units available.

The closest matching container instance for task placement doesn't contain enough CPU units to meet the requirements in the task definition. Review the CPU requirements in both the task size and container definition parameters of the task definition.

service (*service-name*) was unable to place a task because no container instance met all of its requirements. The closest matching container-instance *container-instance-id* encountered error "AGENT".

The Amazon ECS container agent on the closest matching container instance for task placement is disconnected. If you can connect to the container instance with SSH, you can examine the agent logs; for more information, see [Amazon ECS container agent log configuration parameters](#). You should also verify that the agent is running on the instance. If you are using the Amazon ECS-optimized AMI, you can try stopping and restarting the agent with the following command.

- For the Amazon ECS-optimized Amazon Linux 2 AMI and Amazon ECS-optimized Amazon Linux 2023 AMI

```
sudo systemctl restart ecs
```

- For the Amazon ECS-optimized Amazon Linux AMI

```
sudo stop ecs && sudo start ecs
```

service (*service-name*) (task *task-id*) (instance *instance-id*) is unhealthy in (elb *elb-name*) due to (reason Instance has failed at least the UnhealthyThreshold number of health checks consecutively.)

This service is registered with a load balancer and the load balancer health checks are failing. The message includes the task ID to help identify which specific task is failing health checks. For more information, see [Troubleshooting service load balancers in Amazon ECS](#).

service (*service-name*) is unable to consistently start tasks successfully.

This service contains tasks that have failed to start after consecutive attempts. At this point, the service scheduler begins to incrementally increase the time between retries. You should troubleshoot why your tasks are failing to launch. For more information, see [Amazon ECS service throttle logic](#).

After the service is updated, for example with an updated task definition, the service scheduler resumes normal behavior.

service (*service-name*) operations are being throttled. Will try again later.

This service is unable to launch more tasks due to API throttling limits. Once the service scheduler is able to launch more tasks, it will resume.

To request an API rate limit quota increase, open the [AWS Support Center](#) page, sign in if necessary, and choose **Create case**. Choose **Service limit increase**. Complete and submit the form.

service (*service-name*) was unable to stop or start tasks during a deployment because of the service deployment configuration. Update the minimumHealthyPercent or maximumPercent value and try again.

This service is unable to stop or start tasks during a service deployment due to the deployment configuration. The deployment configuration consists of the minimumHealthyPercent and maximumPercent values, which are defined when the service is created. Those values can also be updated on an existing service.

The minimumHealthyPercent represents the lower limit on the number of tasks that should be running for a service during a deployment or when a container instance is draining. It's a percent of the desired number of tasks for the service. This value is rounded up. For example, if the minimum healthy percent is 50 and the desired task count is four, then the scheduler can stop two existing

tasks before starting two new tasks. Likewise, if the minimum healthy percent is 75% and the desired task count is two, then the scheduler can't stop any tasks due to the resulting value also being two.

The `maximumPercent` represents the upper limit on the number of tasks that should be running for a service during a deployment or when a container instance is draining. It's a percent of the desired number of tasks for a service. This value is rounded down. For example, if the maximum percent is 200 and the desired task count is four, then the scheduler can start four new tasks before stopping four existing tasks. Likewise, if the maximum percent is 125 and the desired task count is three, the scheduler can't start any tasks due to the resulting value also being three.

When setting a minimum healthy percent or a maximum percent, you should ensure that the scheduler can stop or start at least one task when a deployment is triggered.

service (*service-name*) was unable to place a task. Reason: You've reached the limit on the number of tasks you can run concurrently

You can request a quota increase for the resource that caused the error. For more information, see [Service quotas](#). To request a quota increase, see [Requesting a quota increase](#) in the *Service Quotas User Guide*.

service (*service-name*) was unable to place a task. Reason: Internal error.

The following is the possible reason for this error:

The service is unable to start a task due to a subnet being in an unsupported Availability Zone.

For information about the supported Fargate Regions and Availability Zones, see [the section called "AWS Fargate Regions"](#).

For information about how to view the subnet Availability Zone, see [View your subnet](#) in the *Amazon VPC User Guide*.

service (*service-name*) was unable to place a task. Reason: The requested CPU configuration is above your limit.

You can request a quota increase for the resource that caused the error. For more information, see [Service quotas](#). To request a quota increase, see [Requesting a quota increase](#) in the *Service Quotas User Guide*.

service (*service-name*) was unable to place a task. Reason: The requested MEMORY configuration is above your limit.

You can request a quota increase for the resource that caused the error. For more information, see [Service quotas](#). To request a quota increase, see [Requesting a quota increase](#) in the *Service Quotas User Guide*.

service (*service-name*) was unable to place a task. Reason: You've reached the limit on the number of vCPUs you can run concurrently

AWS Fargate is transitioning from task count-based quotas to vCPU-based quotas.

You can request a quota increase for Fargate vCPU-based quota. For more information, see [Service quotas](#). To request a Fargate quota increase, see [Requesting a quota increase](#) in the *Service Quotas User Guide*.

service (*service-name*) was unable to reach steady state because task set (*taskSet-ID*) was unable to scale in. Reason: The number of protected tasks are more than the desired count of tasks

The service has more protected tasks than the desired number of tasks. You can do one the following:

- Wait until the protection on the current tasks expire, enabling them to be terminated.
- Determine which tasks can be stopped and use the `UpdateTaskProtection` API with the `protectionEnabled` option set to `false` to unset protection for these tasks.
- Increase the desired task count of the service to more than the number of protected tasks.

service (*service-name*) was unable to reach steady state. Reason: No Container Instances were found in your capacity provider.

The service scheduler sends this event message when it couldn't find the available resources to add another task. The possible causes for this are:

There is no capacity provider associated with the cluster

Use `describe-services` to verify that you have a capacity provider associated with the cluster. You can update the capacity provider strategy for the service.

Verify that there is available capacity in the capacity provider. In the case of EC2, make sure that the container instances meet the task definition requirements.

No container instances were found in your cluster

If no container instances are registered in the cluster you attempt to run a task in, you receive this error. You should add container instances to your cluster. For more information, see [Launching an Amazon ECS Linux container instance](#).

Not enough ports

If your task uses fixed host port mapping (for example, your task uses port 80 on the host for a web server), you must have at least one container instance per task. Only one container can use a single host port at a time. You should add container instances to your cluster or reduce your number of desired tasks.

Too many ports registered

The closest matching container instance for task placement can't exceed the maximum allowed reserved port limit of 100 host ports per container instance. Using dynamic host port mapping may remediate the issue.

Port already in-use

The task definition of this task uses the same port in its port mapping as a task already running on the container instance that was chosen. The service event message would have the chosen container instance ID as part of the message below.

```
The closest matching container-instance is already using a port required by your task.
```

Not enough memory

If your task definition specifies 1000 MiB of memory, and the container instances in your cluster each have 1024 MiB of memory, you can only run one copy of this task per container instance. You can experiment with less memory in your task definition so that you could launch more than one task per container instance, or launch more container instances into your cluster.

Note

If you are trying to maximize your resource utilization by providing your tasks as much memory as possible for a particular instance type, see [Reserving Amazon ECS Linux container instance memory](#).

Not enough available ENI attachment points

Tasks that use the awsvpc network mode each receive their own elastic network interface (ENI), which is attached to the container instance that hosts it. Amazon EC2 instances have a limit to the number of ENIs that can be attached to them, and there are no container instances in the cluster that have ENI capacity available.

The ENI limit for individual container instances depends on the following conditions:

- If you **have not** opted in to the awsvpcTrunking account setting, the ENI limit for each container instance depends on the instance type. For more information, see [IP Addresses Per Network Interface Per Instance Type](#) in the *Amazon EC2 User Guide*.
- If you **have** opted in to the awsvpcTrunking account setting but you **have not** launched new container instances using a supported instance type after opting in, the ENI limit for each container instance is still at the default value. For more information, see [IP Addresses Per Network Interface Per Instance Type](#) in the *Amazon EC2 User Guide*.
- If you **have** opted in to the awsvpcTrunking account setting and you **have** launched new container instances using a supported instance type after opting in, additional ENIs are available. For more information, see [Supported instances for increased Amazon ECS container network interfaces](#).

For more information about opting in to the awsvpcTrunking account setting, see [Increasing Amazon ECS Linux container instance network interfaces](#).

You can add container instances to your cluster to provide more available network adapters.

Container instance missing required attribute

Some task definition parameters require a specific Docker remote API version to be installed on the container instance. Others, such as the logging driver options, require the container instances to register those log drivers with the ECS_AVAILABLE_LOGGING_DRIVERS agent configuration variable. If your task definition contains a parameter that requires a specific

container instance attribute, and you don't have any available container instances that can satisfy this requirement, the task cannot be placed.

A common cause of this error is if your service is using tasks that use the `awsvpc` network mode and EC2 and the cluster you specified doesn't have a container instance registered to it in the same subnet that was specified in the `awsvpcConfiguration` when the service was created.

You can use the `AWSSupport-TroubleshootECSContainerInstance` runbook to troubleshoot. The runbook reviews whether the user data for the instance contains the correct cluster information, whether the instance profile contains the required permissions, and network configuration issues. For more information, see [AWSSupport-TroubleshootECSContainerInstance](#) in the *AWS Systems Manager Automation runbook reference User Guide*.

For more information on which attributes are required for specific task definition parameters and agent configuration variables, see [Amazon ECS task definition parameters for Fargate](#) and [Amazon ECS container agent configuration](#).

service (*service-name*) was unable to place a task. Reason: Capacity is unavailable at this time. Please try again later or in a different availability zone.

There is currently no available capacity to run your service on.

You can do one the following:

- Wait until the Fargate capacity or EC2 container instances become available.
- Relaunch the service and specify additional subnets.

service (*service-name*) deployment failed: tasks failed to start.

The tasks in your service failed to start.

For information about how to debug stopped tasks, see [Amazon ECS stopped tasks error messages](#).

service (*service-name*) Timed out waiting for Amazon ECS Agent to start. Please check logs at `/var/log/ecs/ecs-agent.log`.

The Amazon ECS container agent on the closest matching container instance for task placement is disconnected. If you can connect to the container instance with SSH, you can examine the agent

logs. For more information, see [Amazon ECS container agent log configuration parameters](#). You should also verify that the agent is running on the instance. If you are using the Amazon ECS-optimized AMI, you can try stopping and restarting the agent with the following command.

- For the Amazon ECS-optimized Amazon Linux 2 AMI

```
sudo systemctl restart ecs
```

- For the Amazon ECS-optimized Amazon Linux AMI

```
sudo stop ecs && sudo start ecs
```

service (*service-name*) task set (*taskSet-ID*) (task *task-id*) is not healthy in target-group (*targetGroup-ARN*) due to TARGET GROUP IS NOT FOUND.

The task set for the service is failing health checks because the target group isn't found. The message includes the task ID to help identify which specific task is failing health checks. You should delete and recreate the service. Don't delete any Elastic Load Balancing target group unless the corresponding Amazon ECS service is already deleted.

service (*service-name*) task set (*taskSet-ID*) (task *task-id*) is not healthy in target-group (*targetGroup-ARN*) due to TARGET IS NOT FOUND.

The task set for the service is failing health checks because the target isn't found. The message includes the task ID to help identify which specific task is failing health checks.

IAM permissions policies have been misconfigured or changed, and ECS can no longer maintain your service

The service is unable to maintain tasks due to misconfigured or changed IAM permissions policies. The IAM role associated with your ECS service or tasks may be missing required permissions.

To resolve this issue, add the necessary permissions to the IAM role. For more information about managing IAM permissions policies, see [Adding and removing IAM identity permissions](#) in the *IAM User Guide*.

IAM trust relationship has been misconfigured or changed, and ECS can no longer maintain your service

The service is unable to maintain tasks due to a misconfigured or changed IAM trust relationship. The IAM role associated with your ECS service or tasks may have an incorrect trust policy.

To resolve this issue, configure a trust policy for the role used in your task definition. For more information about creating trust policies for custom roles, see [Creating a role for a custom use case](#) in the *IAM User Guide*.

service (*service-name*) could not launch *number* tasks for deployment *deployment-id*.

The service scheduler sends this event message when a deployment workflow successfully starts some tasks but fails to launch all requested tasks due to insufficient capacity errors. This typically occurs when Circuit Breaker is enabled and provides visibility into why deployments may fail or roll back.

The message includes the specific failure reason, such as insufficient CPU, memory, or other resource constraints. This helps you understand what resources need to be addressed to resolve the deployment issue.

For more information, see [service \(*service-name*\) was unable to place a task because no container instance met all of its requirements..](#)

service (*service-name*) was unable to place tasks in your cluster because the tasks provisioning capacity limit was exceeded.

The service scheduler sends this event message when your cluster has reached the limit of 500 tasks that can be in the PROVISIONING state simultaneously. This is a cluster-level limit, not a service-specific issue.

This typically occurs when you start a service with a high number of desired tasks with limited pre-provisioned capacity, or when multiple services are started simultaneously causing high task churn.

To resolve this issue:

- Wait for existing tasks to complete provisioning and move to the RUNNING state.
- Consider scaling your services more gradually to avoid hitting the provisioning limit.

- Review your cluster's capacity provider configuration to ensure adequate resources are available.

For more information about Amazon ECS service quotas, see [Amazon Elastic Container Service endpoints and quotas](#) in the *Amazon Web Services General Reference*.

Amazon ECS service unhealthy event messages

The following are examples of service unhealthy event messages.

EC2: (service *service-name*) (task *task-id*) (instance *instance-id*) (port *port-number*) is unhealthy in (target-group *target-group-name*) due to (reason *failure-reason*)

This message indicates that a task running on an EC2 instance is failing health checks. For more information, see the following:

- [How do I get my Amazon ECS tasks that use EC2 to pass the Application Load Balancer health check?](#)

EC2 with taskSet: (service *service-name*, taskSet *taskSet-id*) (task *task-id*) (instance *instance-id*) (port *port-number*) is unhealthy in (target-group *target-group-name*) due to (reason *failure-reason*)

This message indicates that a task in a task set running on an EC2 instance is failing health checks. For more information, see the following:

- [How do I get my Amazon ECS tasks that use the EC2 to pass the Application Load Balancer health check?](#)

Fargate: (service *service-name*) (task *task-id*) (port *port-number*) is unhealthy in (target-group *target-group-name*) due to (reason *failure-reason*)

This message indicates that a Fargate task is failing health checks.

For more information about troubleshooting health check failures in Fargate tasks, see [How do I troubleshoot health check failures for Amazon ECS tasks on Fargate?](#).

Fargate with taskSet: (service *service-name*, taskSet *taskSet-id*) (task *task-id*) (port *port-number*) is unhealthy in (target-group *target-group-name*) due to (reason *failure-reason*)

This message indicates that a task in a task set running on Fargate is failing health checks.

For more information about troubleshooting health check failures in Fargate tasks, see [How do I troubleshoot health check failures for Amazon ECS tasks on Fargate?](#).

Amazon ECS Availability Zone service rebalancing service event messages

The following are examples of service event messages you might see.

service (*service-name*) is not AZ balanced with *number-tasks* tasks in *Availability Zone 1*, *number-tasks* in *Availability Zone 2*, and *number-tasks* in *Availability Zone 3*. AZ Rebalancing in progress.

The service scheduler sends a service (*service-name*) is not AZ balanced service event when the number of tasks is not evenly spread across Availability Zones. There is no action to take. This is an information event.

service (*service-name*) is AZ balanced with *number-tasks* tasks in *Availability Zone 1*, *number-tasks* tasks in *Availability Zone 2*, and *number-tasks* tasks in *Availability Zone 3*.

The service scheduler sends a service (*service-name*) is AZ balanced service event when Availability Zone service rebalancing completes. There is no action to take. This is an information event.

***service-name* has started *number-tasks* tasks in *Availability Zone* to AZ Rebalance: *task-ids*.**

The service scheduler sends a *service-name/task-set-name* has started *number* tasks in *Availability Zone* service event when it starts tasks in an Availability Zone because of service rebalancing. There is no action to take. This is an information event.

***service-name* has stopped *number-tasks* running tasks in *Availability Zone* due to AZ rebalancing: *task-id*.**

The service scheduler sends a *service-name/task-set-name* has stopped *number* tasks in *Availability Zone* service event when it stops tasks in an Availability Zone because of service rebalancing. There is no action to take. This is an information event.

service (*service-name*) is unable to place a task in *Availability Zone* because no container instance met all of its requirements.

The service scheduler sends a *service-name* is unable to place a task in *Availability Zone* service event because no container instance met all of its requirements. To address the issue, launch instances in the Availability Zone.

service (*service-name*) is unable to place a task in *Availability Zone*.

The service scheduler sends a *service-name* is unable to place a task in *Availability Zone* service event when you use the Fargate and there is no available capacity.

You can add additional subnets in the Availability Zone in the error message, or contact Support to get additional capacity.

service (*service-name*) was unable to AZ Rebalance because *task-set-name* was unable to scale in due to *reason*.

The service scheduler sends a *service-name* was unable to AZ Rebalance because *task-set-name* was unable to scale in due to *reason* service event when you use task scale-in protection.

You can do one the following:

- Wait until the protection on the current tasks expire, enabling them to be terminated.
- Determine which tasks can be stopped and use the UpdateTaskProtection API with the protectionEnabled option set to false to unset protection for these tasks.
- Increase the desired task count of the service to more than the number of protected tasks.

service (*service-name*) stopped AZ Rebalancing.

The service scheduler sends a *service-name* stopped AZ Rebalancing service event when the Availability Zone rebalancing operation stopped. This is an information event. Amazon ECS sends additional events which provide more information.

Troubleshooting service load balancers in Amazon ECS

Amazon ECS services can register tasks with an Elastic Load Balancing load balancer. Load balancer configuration errors are common causes for stopped tasks. If your stopped tasks were started by services that use a load balancer, consider the following possible causes.

Amazon ECS service-linked role doesn't exist

The Amazon ECS service-linked role allows Amazon ECS services to register container instances with Elastic Load Balancing load balancers. The service-linked role must be created in your account. For more information, see [Using service-linked roles for Amazon ECS](#).

Container instance security group

If your container is mapped to port 80 on your container instance, your container instance security group must allow inbound traffic on port 80 for the load balancer health checks to pass.

Elastic Load Balancing load balancer not configured for all Availability Zones

Your load balancer should be configured to use all of the Availability Zones in a Region, or at least all of the Availability Zones where your container instances reside. If a service uses a load balancer and starts a task on a container instance that resides in an Availability Zone that the load balancer isn't configured to use, the task never passes the health check. This results in the task being killed.

Elastic Load Balancing load balancer health check misconfigured

The load balancer health check parameters can be overly restrictive or point to resources that don't exist. If a container instance is determined to be unhealthy, it's removed from the load balancer. Be sure to verify that the following parameters are configured correctly for your service load balancer.

Ping Port

The **Ping Port** value for a load balancer health check is the port on the container instances that the load balancer checks to determine if it is healthy. If this port is misconfigured, the

load balancer likely deregisters your container instance from itself. This port should be configured to use the `hostPort` value for the container in your service's task definition that you're using with the health check.

Ping Path

This is part of the load balancer healthcheck. It is an endpoint on your application that can return a successful status code (for example, 200) when the application is healthy. This value is often set to `index.html`, but if your service doesn't respond to that request, then the health check fails. If your container doesn't have an `index.html` file, you can set this to `/` to target the base URL for the container instance.

Response Timeout

This is the amount of time that your container has to return a response to the health check ping. If this value is lower than the amount of time required for a response, the health check fails.

Health Check Interval

This is the amount of time between health check pings. The shorter your health check intervals are, the faster your container instance can reach the **Unhealthy Threshold**.

Unhealthy Threshold

This is the number of times your health check can fail before your container instance is considered unhealthy. If you have an unhealthy threshold of 2, and a health check interval of 30 seconds, then your task has 60 seconds to respond to the health check ping before it's assumed unhealthy. You can raise the unhealthy threshold or the health check interval to give your tasks more time to respond.

Unable to update the service *servicename*: Load balancer container name or port changed in task definition

If your service uses a load balancer, you can use the AWS CLI or SDK to modify the load balancer configuration. For information about how to modify the configuration, see [UpdateService](#) in the *Amazon Elastic Container Service API Reference*. If you update the task definition for the service, the container name and container port that are specified in the load balancer configuration must remain in the task definition.

You've reached the limit on the number of tasks that you can run concurrently.

For a new account, your quotas might be lower than the service quotas. The service quota for your account can be viewed in the Service Quotas console. To request a quota increase, see [Requesting a quota increase](#) in the *Service Quotas User Guide*.

Troubleshooting service auto scaling in Amazon ECS

Application Auto Scaling turns off scale-in processes while Amazon ECS deployments are in progress, and they resume once the deployment has completed. However, scale-out processes continue to occur, unless suspended, during a deployment. For more information, see [Suspending and resuming scaling for Application Auto Scaling](#).

Amazon ECS event capture in the console

The Amazon ECS console provides event capture functionality that stores Amazon ECS-generated events, such as service actions and task state changes, to Amazon CloudWatch Logs through EventBridge. This feature includes a query interface with filtering capabilities for monitoring and troubleshooting.

Events provide detailed information about how your service deployments, services, tasks, and instances operate. You can use this information to troubleshoot task or service deployment failures.

When you turn on event capture, you have access to all events Amazon ECS generates for a retention period of your choice, extending beyond the native limitations of the last 100 unfiltered events or stopped tasks visible for only 1 hour.

How it works

Event capture uses EventBridge to store events in a predefined Amazon CloudWatch Logs log group. The Amazon ECS console provides pre-built queries and filtering options, and correlates events to provide task lifecycles in an intuitive format.

You can query and retrieve the following types of events:

- **Service action events** – Help identify provisioning or resource allocation issues
- **Task lifecycle events** – Help identify why tasks or containers fail to launch or stop running

The Amazon ECS console allows you to set up event capture in one click and provides commonly used queries and filtering without requiring you to learn query languages or navigate between multiple consoles.

Event types

Event capture stores all Amazon ECS generated events in the following categories:

Task state change events

Container stops and other termination events, which you can use for troubleshooting or to monitor task lifecycle timelines.

Service actions

Events such as reaching steady state, failed task placement, or resource constraints.

Service deployment state changes

Events such as in-progress, completed, or failed deployments, triggered by circuit breaker and rollback settings, to monitor the state of a service deployment.

Container instance state changes

For workloads on EC2 and Amazon ECS Managed Instances, events show connected and disconnected status.

Log group configuration

When you turn on event capture, Amazon ECS automatically creates the following resources:

- A Amazon CloudWatch Logs log group named `/aws/events/ecs/containerinsights/${clusterName}/performance`
- An EventBridge rule that ingests all events from the `aws.ecs` source and forwards them to the log group

You can specify a retention period for the log group from 1 day to 10 years. The default retention period is 7 days.

Considerations

Consider the following when using event capture:

- Event capture stores all events for simplicity. You cannot configure rules in the Amazon ECS console to capture only specific events.
- The Amazon ECS console provides predefined query criteria. For advanced queries, use Amazon CloudWatch Logs Logs Insights to query the log group directly.
- Live tail functionality is not available in the Amazon ECS console. Use Amazon CloudWatch Logs directly for live tail.
- When you disable event capture, the EventBridge rule is deleted.
- Event capture incurs additional costs for EventBridge data ingestion, Amazon CloudWatch Logs storage, and query execution.

For information about EventBridge pricing, see [EventBridge pricing](#).

For information about CloudWatch pricing, see [CloudWatch pricing](#).

Event-based troubleshooting

Use Amazon ECS generated events to answer common troubleshooting questions.

Task failure analysis

You can review STOPPED task state change events, stop codes, and container exit codes to determine why a task failed to launch or failed while running.

You can review service action events for placement failures and resource constraint information to determine why a task failed to place due to resource constraints

Common task failure scenarios

The most common abnormal task failures are related to the following issues:

- CI/CD service deployment failures
- Auto scaling failures
- Task rebalancing failures
- Abnormal container exits, such as out-of-memory (OOM) errors

Abnormal task failures produce STOPPED task state change events with an `EssentialContainerExited` or `TaskFailedToStart` stop code. You can filter by these stop codes to examine container execution and stopping behaviors.

Turn on event capture for an existing Amazon ECS cluster

You can enable event capture on an existing Amazon ECS cluster to store Amazon ECS-generated events in Amazon CloudWatch Logs through EventBridge. This feature helps you monitor and troubleshoot task failures, service deployments, and other cluster activities.

After you enable event capture, Amazon ECS creates the following resources:

- A Amazon CloudWatch Logs log group named `/aws/events/ecs/containerinsights/{clusterName}/performance`
- An EventBridge rule that captures all events from the `aws.ecs` source

A **History** tab displays in the cluster view, allowing you to query task lifecycle events and service actions. Event capture begins immediately and stores all Amazon ECS-generated events according to your specified retention period.

Prerequisites

- An existing Amazon ECS cluster
- Appropriate IAM permissions to modify cluster settings and create Amazon CloudWatch Logs resources

Turn on event capture using the console

1. Open the console at <https://console.aws.amazon.com/ecs/v2>.
2. In the navigation pane, choose **Clusters**.
3. Select the cluster where you want to enable event capture.

The cluster details page displays.

4. Choose **Configuration**.
5. In the **ECS events** section, choose **Turn on event capture**.

The **Turn on event capture** dialog box displays.

6. For **Expire event**, choose the retention period for the Amazon CloudWatch Logs log group. The default is 7 days.
7. Choose **Turn on**.

Viewing Amazon ECS service and task state change events

The Amazon ECS console provides event capture functionality that stores Amazon ECS-generated events, such as service actions and task state changes, to Amazon CloudWatch Logs through EventBridge. This feature includes a query interface with filtering capabilities to enhance monitoring and troubleshooting.

Events provide detailed information about how your service deployments, services, tasks, and instances operate. You can use this information to troubleshoot task or service deployment failures.

You can use any of the following criteria to filter the events:

- Deployment ID (This is only available on the service detail page)
- Start time
- End time
- Service name (only applicable on cluster detail page, on service detail page, this will be default to current service)
- Task ID
- Task Last status
- Task definition family
- Task definition revision

Viewing events at the cluster-level

1. Open the console at <https://console.aws.amazon.com/ecs/v2>.
2. Choose **Clusters**.

The clusters list page displays.

3. Choose the cluster.

The cluster details page displays.

4. Under **History**, determine the events to view.
 - a. To view service action events, choose **Service action events**.
 - b. To view task state change events, choose **Task state change events**.
 - c. (Optional) In **Query criteria**, enter the filters for the events that you want to view.
5. Choose **Run query**.

The events display in a list.

6. To view the full details of the event, choose the event.

Viewing at the service-level

1. Open the console at <https://console.aws.amazon.com/ecs/v2>.
2. On the **Clusters** page, choose the cluster.
3. On the cluster details page, in the **Services** section, choose the service.

The service details page displays.

4. Under **History**, determine the events to view.
 - a. To view service action events, choose **Service action events**.
 - b. To view task state change events, choose **Task state change events**.
 - c. (Optional) In **Query criteria**, enter the filters for the events that you want to view.
5. Choose **Run query**.

The events display in a list.

6. To view the full details of the event, choose the event.

Troubleshoot Amazon ECS task definition invalid CPU or memory errors

When registering a task definition using the Amazon ECS API or AWS CLI, if you specify an invalid cpu or memory value, the following error is returned.

```
An error occurred (ClientException) when calling the RegisterTaskDefinition operation:  
Invalid 'cpu' setting for task.
```

Note

When using Terraform, the following error might be returned.



```
Error: ClientException: No Fargate configuration exists for given values.
```

To resolve this issue, you must specify a supported value for the task CPU and memory in your task definition. The `cpu` value can be expressed in CPU units or vCPUs in a task definition. It's converted to an integer indicating the CPU units when the task definition is registered. The memory value can be expressed in MiB or GB in a task definition. It's converted to an integer indicating the MiB when the task definition is registered.

For task definitions that specify `FARGATE` for the `requiresCompatibilities` parameter (even if `EC2` is also specified), you must use one of the values in the following table. These values determine your range of supported values for the CPU and memory parameter.

For tasks hosted on Fargate, the following table shows the valid CPU and memory combinations. The memory values in the JSON file are specified in MiB. You can convert the GB value to MiB by multiplying the value by 1024. For example 1 GB = 1024 MiB.

CPU value	Memory value	Operating systems supported for AWS Fargate
256 (.25 vCPU)	512 MiB, 1 GB, 2 GB	Linux
512 (.5 vCPU)	1 GB, 2 GB, 3 GB, 4 GB	Linux
1024 (1 vCPU)	2 GB, 3 GB, 4 GB, 5 GB, 6 GB, 7 GB, 8 GB	Linux, Windows
2048 (2 vCPU)	Between 4 GB and 16 GB in 1 GB increments	Linux, Windows
4096 (4 vCPU)	Between 8 GB and 30 GB in 1 GB increments	Linux, Windows
8192 (8 vCPU)	Between 16 GB and 60 GB in 4 GB increments	Linux

CPU value	Memory value	Operating systems supported for AWS Fargate
<div>  Note This option requires Linux platform 1.4.0 or later. </div>		
16384 (16vCPU)	Between 32 GB and 120 GB in 8 GB increments	Linux
<div>  Note This option requires Linux platform 1.4.0 or later. </div>		

For tasks hosted on Amazon EC2, supported task CPU values are between 0.25 vCPUs and 192 vCPUs.

The CPU control mechanism differs between EC2 and Fargate:

- For tasks hosted on Amazon EC2: Amazon ECS uses the CPU period and the CPU quota to control the task size CPU hard limits. When you specify the vCPU in your task definition, Amazon ECS translates the value to the CPU period and CPU quota settings that apply to the cgroup.
- For tasks hosted on Fargate: Amazon ECS uses CPU shares to control CPU allocation. The CPU quota and period values are not used for CPU limiting in Fargate tasks.

For Amazon EC2 tasks, the CPU quota controls the amount of CPU time granted to a cgroup during a given CPU period. Both settings are expressed in terms of microseconds. When the CPU quota equals the CPU period means a cgroup can execute up to 100% on one vCPU (or any other fraction that totals to 100% for multiple vCPUs). The CPU quota has a maximum of 1000000us and the CPU period has a minimum of 1ms. You can use these values to set the limits for your CPU count. When you change the CPU period without changing the CPU quota, you have different effective limits than what you've specified in your task definition.

The 100ms period allows for vCPUs ranging from 0.125 to 10.

 **Note**

Task-level CPU and memory parameters are ignored for Windows containers.

Viewing Amazon ECS container agent logs

Amazon ECS stores logs in the `/var/log/ecs` folder of your container instances. There are logs available from the Amazon ECS container agent and from the `ecs-init` service that controls the state of the agent (start/stop) on the container instance. You can view these log files by connecting to a container instance using SSH.

 **Note**

If you are not sure how to collect all of the logs on your container instances, you can use the Amazon ECS logs collector. For more information, see [Collecting container logs with Amazon ECS logs collector](#).

Linux operating system

The `ecs-init` process stores logs at `/var/log/ecs/ecs-init.log`.

The `ecs-init.log` file contains information about the container agent lifecycle management, configuration, and bootstrapping.

You can use the following command to view the log files.

```
cat /var/log/ecs/ecs-init.log
```

Output:

```
2018-02-16T18:13:54Z [INFO] pre-start
2018-02-16T18:13:56Z [INFO] start
2018-02-16T18:13:56Z [INFO] No existing agent container to remove.
2018-02-16T18:13:56Z [INFO] Starting Amazon Elastic Container Service Agent
```

Windows operating system

You can use the Amazon ECS logs collector for Windows. For more information, see [Amazon ECS Logs Collector For Windows](#) on Github.

1. Connect to your instance.
2. Open PowerShell and then run the following commands with administrative privileges. The commands download the script and collect the logs.

```
Invoke-WebRequest -OutFile ecs-logs-collector.ps1 https://  
raw.githubusercontent.com/aws-labs/aws-ecs-logs-collector-for-windows/master/ecs-  
logs-collector.ps1  
.\ecs-logs-collector.ps1
```

You can turn on debug logging for Amazon ECS agent and the Docker daemon. This option allows the script to collect the logs before turning on debug mode. The script restarts the Docker daemon and Amazon ECS agent, and then terminates all containers running on the instance. Before running the following command, drain the container instance and moving any important tasks to other container instances.

Run the following command to turn on logging.

```
.\ecs-logs-collector.ps1 -RunMode debug
```

Collecting container logs with Amazon ECS logs collector

Note

You can't use the Amazon ECS logs collector on Amazon ECS Managed Instances.

If you are unsure how to collect all of the various logs on your container instances, you can use the Amazon ECS logs collector. It is available on GitHub for both [Linux](#) and [Windows](#). The script collects general operating system logs as well as Docker and Amazon ECS container agent logs, which can be helpful for troubleshooting AWS Support cases. It then compresses and archives the collected information into a single file that can easily be shared for diagnostic purposes. It also supports

enabling debug mode for the Docker daemon and the Amazon ECS container agent on Amazon Linux variants, such as the Amazon ECS-optimized AMI.

 **Note**

On Amazon Linux Amazon ECS-optimized AMIs version 20250909 and later, the Amazon ECS logs collector is pre-installed at `/opt/amazon/ecs/ecs-logs-collector.sh` and ready to use without downloading from GitHub. For more information, see [ECS Logs Collector](#) in the ECS-optimized AMI documentation.

Currently, the Amazon ECS logs collector supports the following operating systems:

- Amazon Linux
- Red Hat Enterprise Linux
- Ubuntu
- Debian
- Windows Server

 **Note**

The source code for the Amazon ECS logs collector is available on GitHub for both [Linux](#) and [Windows](#). We encourage you to submit pull requests for changes that you would like to have included. However, Amazon Web Services doesn't currently support running modified copies of this software.

To download and run the Amazon ECS logs collector for Linux

1. Connect to your container instance.
2. Download the Amazon ECS logs collector script.

```
curl -O https://raw.githubusercontent.com/awslabs/ecs-logs-collector/master/ecs-logs-collector.sh
```

3. Run the script to collect the logs and create the archive.

 **Note**

To enable the debug mode for the Docker daemon and the Amazon ECS container agent, add the `--mode=enable-debug` option to the following command. This might restart the Docker daemon, which kills all containers that are running on the instance. Consider draining the container instance and moving any important tasks to other container instances before enabling debug mode. For more information, see [Draining Amazon ECS container instances](#).

```
[ec2-user ~]$ sudo bash ./ecs-logs-collector.sh
```

 **Important**

We recommend that you edit the logs and remove all sensitive data from the files. You can search for known data, and also search for environment variables such as `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, and `AWS_SESSION_TOKEN` in the file.

After you have run the script, you can examine the collected logs in the `collect` folder that the script created. The `collect.tgz` file is a compressed archive of all of the logs, which you can share with AWS Support for diagnostic help.

To download and run the Amazon ECS logs collector for Windows

1. Connect to your container instance. For more information, see [Connect to your Windows instance using RDP](#) in the *Amazon EC2 User Guide*.
2. Download the Amazon ECS logs collector script using PowerShell.

```
Invoke-WebRequest -OutFile ecs-logs-collector.ps1 https://raw.githubusercontent.com/aws-labs/aws-ecs-logs-collector-for-windows/master/ecs-logs-collector.ps1
```

3. Run the script to collect the logs and create the archive.

 **Note**

To enable the debug mode for the Docker daemon and the Amazon ECS container agent, add the `-RunMode debug` option to the following command. This restarts the Docker daemon, which kills all containers that are running on the instance. Consider draining the container instance and moving any important tasks to other container instances before enabling debug mode. For more information, see [Draining Amazon ECS container instances](#).

```
.\ecs-logs-collector.ps1
```

 **Important**

We recommend that you edit the logs and remove all sensitive data from the files. You can search for known data, and also search for environment variables such as `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, and `AWS_SESSION_TOKEN` in the file.

After you have run the script, you can examine the collected logs in the `collect` folder that the script created. The `collect.tgz` file is a compressed archive of all of the logs, which you can share with AWS Support for diagnostic help.

Retrieve Amazon ECS diagnostic details with agent introspection

The Amazon ECS agent introspection API provides information about the overall state of the Amazon ECS agent and the container instances.

You can use the agent introspection API to get the Docker ID for a container in your task. You can use the agent introspection API by connecting to a container instance using SSH.

⚠ Important

Your container instance must have an IAM role that allows access to Amazon ECS in order to reach the introspection API. For more information, see [Amazon ECS container instance IAM role](#).

The following example shows two tasks, one that is currently running and one that was stopped.

📌 Note

The following command is piped through the **python -mjson.tool** for greater readability.

```
curl http://localhost:51678/v1/tasks | python -mjson.tool
```

Output:

```
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                               Dload  Upload   Total   Spent    Left   Speed
100 1095    100 1095     0     0  117k      0 --:--:-- --:--:-- --:--:-- 133k
{
  "Tasks": [
    {
      "Arn": "arn:aws:ecs:us-west-2:aws_account_id:task/090eff9b-1ce3-4db6-848a-a8d14064fd24",
      "Containers": [
        {
          "DockerId":
            "189a8ff4b5f04affe40e5160a5ffadca395136eb5faf4950c57963c06f82c76d",
          "DockerName": "ecs-console-sample-app-static-6-simple-app-86caf9bcabe3e9c61600",
          "Name": "simple-app"
        },
        {
          "DockerId":
            "f7f1f8a7a245c5da83aa92729bd28c6bcb004d1f6a35409e4207e1d34030e966",
          "DockerName": "ecs-console-sample-app-static-6-busybox-ce83ce978a87a890ab01",
          "Name": "busybox"
        }
      ]
    }
  ]
}
```

```

    ],
    "Family": "console-sample-app-static",
    "KnownStatus": "STOPPED",
    "Version": "6"
  },
  {
    "Arn": "arn:aws:ecs:us-west-2:aws_account_id:task/1810e302-eaea-4da9-
a638-097bea534740",
    "Containers": [
      {
        "DockerId":
"dc7240fe892ab233dbbcee5044d95e1456c120dba9a6b56ec513da45c38e3aeb",
        "DockerName": "ecs-console-sample-app-static-6-simple-app-
f0e5859699a7aecfb101",
        "Name": "simple-app"
      },
      {
        "DockerId":
"096d685fb85a1ff3e021c8254672ab8497e3c13986b9cf005cbae9460b7b901e",
        "DockerName": "ecs-console-sample-app-static-6-
busybox-92e4b8d0ecd0cce69a01",
        "Name": "busybox"
      }
    ],
    "DesiredStatus": "RUNNING",
    "Family": "console-sample-app-static",
    "KnownStatus": "RUNNING",
    "Version": "6"
  }
]
}

```

In the preceding example, the stopped task (*090eff9b-1ce3-4db6-848a-a8d14064fd24*) has two containers. You can use **docker inspect *container-ID*** to view detailed information on each container. For more information, see [Amazon ECS container introspection](#).

Docker diagnostics in Amazon ECS

Docker provides several diagnostic tools that help you troubleshoot problems with your containers and tasks. For more information about all of the available Docker command line utilities, see the [Docker CLI reference](#) in the Docker documentation. You can access the Docker command line utilities by connecting to a container instance using SSH.

The exit codes that Docker containers report can also provide some diagnostic information (for example, exit code 137 means that the container received a SIGKILL signal). For more information, see [Exit Status](#) in the Docker documentation.

List Docker containers in Amazon ECS

You can use the **docker ps** command on your container instance to list the running containers. In the following example, only the Amazon ECS container agent is running. For more information, see [docker ps](#) in the Docker documentation.

```
docker ps
```

Output:

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS	NAMES	
cee0d6986de0	amazon/amazon-ecs-agent:latest	"/agent"	22 hours ago
Up 22 hours	127.0.0.1:51678->51678/tcp	ecs-agent	

You can use the **docker ps -a** command to see all containers (even stopped or killed containers). This is helpful for listing containers that are unexpectedly stopping. In the following example, container f7f1f8a7a245 exited 9 seconds ago, so it doesn't show up in a **docker ps** output without the -a flag.

```
docker ps -a
```

Output:

CONTAINER ID	IMAGE	COMMAND	
CREATED	STATUS	PORTS	NAMES
db4d48e411b1	amazon/ecs-emptyvolume-base:autogenerated	"not-applicable"	
19 seconds ago			ecs-
console-sample-app-static-6-internalecs-emptyvolume-source-c09288a6b0cba8a53700			
f7f1f8a7a245	busybox:buildroot-2014.02	"\sh -c '/bin/sh -c	
22 hours ago	Exited (137) 9 seconds ago		ecs-
console-sample-app-static-6-busybox-ce83ce978a87a890ab01			
189a8ff4b5f0	httpd:2	"httpd-foreground"	
22 hours ago	Exited (137) 40 seconds ago		ecs-
console-sample-app-static-6-simple-app-86caf9bcabe3e9c61600			

```
0c7dca9321e3      amazon/ecs-emptyvolume-base:autogenerated      "not-applicable"
  22 hours ago                                     ecs-
console-sample-app-static-6-internalecs-emptyvolume-source-90fefaa68498a8a80700
cee0d6986de0      amazon/amazon-ecs-agent:latest                  "/agent"
  22 hours ago      Up 22 hours                    127.0.0.1:51678->51678/tcp    ecs-
agent
```

View Docker Logs in Amazon ECS

You can view the STDOUT and STDERR streams for a container with the **docker logs** command. In this example, the logs are displayed for the **dc7240fe892a** container and piped through the **head** command for brevity. For more information, go to [docker logs](#) in the Docker documentation.

Note

Docker logs are only available on the container instance if you are using the default json log driver. If you have configured your tasks to use the awslogs log driver, then your container logs are available in CloudWatch Logs. For more information, see [Send Amazon ECS logs to CloudWatch](#).

```
docker logs dc7240fe892a | head
```

Output:

```
AH00558: httpd: Could not reliably determine the server's fully qualified domain name,
  using 172.17.0.11. Set the 'ServerName' directive globally to suppress this message
AH00558: httpd: Could not reliably determine the server's fully qualified domain name,
  using 172.17.0.11. Set the 'ServerName' directive globally to suppress this message
[Thu Apr 23 19:48:36.956682 2015] [mpm_event:notice] [pid 1:tid 140327115417472]
  AH00489: Apache/2.4.12 (Unix) configured -- resuming normal operations
[Thu Apr 23 19:48:36.956827 2015] [core:notice] [pid 1:tid 140327115417472] AH00094:
  Command line: 'httpd -D FOREGROUND'
10.0.1.86 - - [23/Apr/2015:19:48:59 +0000] "GET / HTTP/1.1" 200 348
10.0.0.154 - - [23/Apr/2015:19:48:59 +0000] "GET / HTTP/1.1" 200 348
10.0.1.86 - - [23/Apr/2015:19:49:28 +0000] "GET / HTTP/1.1" 200 348
10.0.0.154 - - [23/Apr/2015:19:49:29 +0000] "GET / HTTP/1.1" 200 348
10.0.1.86 - - [23/Apr/2015:19:49:50 +0000] "-" 408 -
10.0.0.154 - - [23/Apr/2015:19:49:50 +0000] "-" 408 -
10.0.1.86 - - [23/Apr/2015:19:49:58 +0000] "GET / HTTP/1.1" 200 348
```

```
10.0.0.154 - - [23/Apr/2015:19:49:59 +0000] "GET / HTTP/1.1" 200 348
10.0.1.86 - - [23/Apr/2015:19:50:28 +0000] "GET / HTTP/1.1" 200 348
10.0.0.154 - - [23/Apr/2015:19:50:29 +0000] "GET / HTTP/1.1" 200 348
time="2015-04-23T20:11:20Z" level="fatal" msg="write /dev/stdout: broken pipe"
```

Inspect Docker Containers in Amazon ECS

If you have the Docker ID of a container, you can inspect it with the **docker inspect** command. Inspecting containers provides the most detailed view of the environment in which a container was launched. For more information, see [docker inspect](#) in the Docker documentation.

```
docker inspect dc7240fe892a
```

Output:

```
[{
  "AppArmorProfile": "",
  "Args": [],
  "Config": {
    "AttachStderr": false,
    "AttachStdin": false,
    "AttachStdout": false,
    "Cmd": [
      "httpd-foreground"
    ],
    "CpuShares": 10,
    "Cpuset": "",
    "Domainname": "",
    "Entrypoint": null,
    "Env": [
      "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/
local/apache2/bin",
      "HTTPD_PREFIX=/usr/local/apache2",
      "HTTPD_VERSION=2.4.12",
      "HTTPD_BZ2_URL=https://www.apache.org/dist/httpd/httpd-2.4.12.tar.bz2"
    ],
    "ExposedPorts": {
      "80/tcp": {}
    },
    "Hostname": "dc7240fe892a",
    ...
  }
}]
```

Configuring verbose output from the Docker daemon in Amazon ECS

If you're having trouble with Docker containers or images, you can turn on debug mode on your Docker daemon. Using debugging provides more verbose output from the daemon. You can use this to retrieve error messages that are sent from container registries, such as Amazon ECR.

Important

This procedure is written for the Amazon ECS-optimized Amazon Linux AMI. For other operating systems, see [Enable debugging](#) and [Control and configure Docker with systemd](#) in the Docker documentation.

To use Docker daemon debug mode on the Amazon ECS-optimized Amazon Linux AMI

1. Connect to your container instance.
2. Open the Docker options file with a text editor, such as **vi**. For the Amazon ECS-optimized Amazon Linux AMI, the Docker options file is at `/etc/sysconfig/docker`.
3. Find the Docker options statement and add the `-D` option to the string, inside the quotes.

Note

If the Docker options statement begins with a `#`, remove that character to uncomment the statement and enable the options.

For the Amazon ECS-optimized Amazon Linux AMI, the Docker options statement is called `OPTIONS`. For example:

```
# Additional startup options for the Docker daemon, for example:
# OPTIONS="--ip-forward=true --iptables=true"
# By default we limit the number of open files per container
OPTIONS="-D --default-ulimit nofile=1024:4096"
```

4. Save the file and exit your text editor.
5. Restart the Docker daemon.

```
sudo service docker restart
```

The output is as follows:

```
Stopping docker: [ OK ]
Starting docker: [ OK ]
```

6. Restart the Amazon ECS agent.

```
sudo service ecs restart
```

Your Docker logs should now show more verbose output.

```
time="2015-12-30T21:48:21.907640838Z" level=debug msg="Unexpected response from
server: \"{\\\"errors\\\": [{\\\"code\\\": \\\"DENIED\\\", \\\"message\\\": \\\"User:
arn:aws:sts::1111:assumed-role/ecrReadOnly/i-abcdefg is not authorized to perform:
ecr:InitiateLayerUpload on resource: arn:aws:ecr:us-east-1:1111:repository/nginx_test
\\\"}]}\\n\" http.Header{\\\"Connection\\\": []string{\\\"keep-alive\\\"}, \\\"Content-Type\\\":
[]string{\\\"application/json; charset=utf-8\\\"}, \\\"Date\\\": []string{\\\"Wed, 30 Dec 2015
21:48:21 GMT\\\"}, \\\"Docker-Distribution-API-Version\\\": []string{\\\"registry/2.0\\\"},
\\\"Content-Length\\\": []string{\\\"235\\\"}}\"
```

Troubleshoot the Docker API error (500): devmapper in Amazon ECS

The following Docker error indicates that the thin pool storage on your container instance is full, and that the Docker daemon cannot create new containers:

```
CannotCreateContainerError: API error (500): devmapper: Thin Pool has 4350 free data
blocks which is less than minimum required 4454 free data blocks. Create more free
space in thin pool or use dm.min_free_space option to change behavior
```

By default, Amazon ECS-optimized Amazon Linux AMIs from version 2015.09.d and later launch with an 8-GiB volume for the operating system that's attached at `/dev/xvda` and mounted as the root of the file system. There's an additional 22-GiB volume that's attached at `/dev/xvdcz` that Docker uses for image and metadata storage. If this storage space is filled up, the Docker daemon cannot create new containers.

The easiest way to add storage to your container instances is to terminate the existing instances and launch new ones with larger data storage volumes. However, if you can't do this, you can add storage to the volume group that Docker uses and extend its logical volume by following the procedures in [Amazon ECS-optimized Linux AMIs](#).

If your container instance storage is filling up too quickly, there are a few actions that you can take to reduce this effect:

- To view the thin poll information, run the following command on your container instance:

```
docker info
```

- (Amazon ECS container agent 1.8.0 and later) You can reduce the amount of time that stopped or exited containers remain on your container instances. The `ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION` agent configuration variable sets the time duration to wait from when a task is stopped until the Docker container is removed (by default, this value is 3 hours). This removes the Docker container data. If this value is set too low, you might not be able to inspect your stopped containers or view the logs before they are removed. For more information, see [Amazon ECS container agent configuration](#).
- You can remove non-running containers and unused images from your container instances. You can use the following example commands to manually remove stopped containers and unused images. Deleted containers can't be inspected later, and deleted images must be pulled again before starting new containers from them.

To remove non-running containers, run the following command on your container instance:

```
docker rm $(docker ps -aq)
```

To remove unused images, run the following command on your container instance:

```
docker rmi $(docker images -q)
```

- You can remove unused data blocks within containers. You can use the following command to run **fstrim** on any running container and discard any data blocks that are unused by the container file system.

```
sudo sh -c "docker ps -q | xargs docker inspect --format='{{ .State.Pid }}' | xargs -I Z fstrim /proc/Z/root/"
```

Troubleshoot Amazon ECS Exec issues

The following are troubleshooting notes to help diagnose why you may be getting an error when using ECS Exec.

Verify using the Exec Checker

The ECS Exec Checker script provides a way to verify and validate that your Amazon ECS cluster and task have met the prerequisites for using the ECS Exec feature. The ECS Exec Checker script verifies both your AWS CLI environment and cluster and tasks are ready for ECS Exec, by calling various APIs on your behalf. The tool requires the latest version of the AWS CLI and that the `jq` is available. For more information, see [ECS Exec Checker](#) on GitHub.

Error when calling `execute-command`

If a `The execute command failed` error occurs, the following are possible causes.

- The task does not have the required permissions. Verify that the task definition used to launch your task has a task IAM role defined and that the role has the required permissions. For more information, see [ECS Exec permissions](#).
- The SSM agent isn't installed or isn't running.
- There is an interface Amazon VPC endpoint for Amazon ECS, but there isn't one for Systems Manager Session Manager.

Troubleshoot Amazon ECS Anywhere issues

Amazon ECS Anywhere provides support for registering an *external instance* such as an on-premises server or virtual machine (VM) to your Amazon ECS cluster. The following are common issues that you might encounter and general troubleshooting recommendations for them.

Topics

- [External instance registration issues](#)
- [External instance network issues](#)
- [Issues running tasks on your external instance](#)

External instance registration issues

When registering an external instance with your Amazon ECS cluster, the following requirements must be met:

- An AWS Systems Manager activation, which consists of an *activation ID* and *activation code*, must be retrieved. You use it to register the external instance as a Systems Manager managed instance. When a Systems Manager activation is requested, specify a registration limit and expiration date. The registration limit specifies the maximum number of instances that can be registered using the activation. The default value for registration limit is 1 instance. The expiration date specifies when the activation expires. The default value is 24 hours. If the Systems Manager activation that you're using to register your external instance isn't valid, request a new one. For more information, see [Registering an external instance to an Amazon ECS cluster](#).
- An IAM policy is used to provide your external instance the permissions that it needs to communicate with AWS API operations. If this managed policy isn't created properly and doesn't contain the required permissions, external instance registration fails. For more information, see [Amazon ECS Anywhere IAM role](#).
- Amazon ECS provides an installation script that installs Docker, the Amazon ECS container agent, and the Systems Manager Agent on your external instance. If the installation script fails, it's likely that the script can't be run again on the same instance without an error occurring. If this happens, follow the cleanup process to clear AWS resources from the instance so you can run the installation script again. For more information, see [Deregistering an Amazon ECS external instance](#).

Note

Be aware that, if the installation script successfully requested and used the Systems Manager activation, running the installation script a second time uses the Systems Manager activation again. This might in turn cause you to reach the registration limit for that activation. If this limit is reached, you must create a new activation.

- When running the installation script on an external instance for GPU workloads, if the NVIDIA driver is not detected or configured properly, an error will occur. The installation script uses the `nvidia-smi` command to confirm the existence of the NVIDIA driver.

External instance network issues

To communicate any changes, your external instance requires a network connection to AWS. If your external instance loses its network connection to AWS, tasks that are running on your instances continue to run anyway unless stopped manually. After the connection to AWS is restored, the AWS credentials that are used by the Amazon ECS container agent and Systems Manager Agent on the external instance renew automatically. For more information about the AWS domains that are used for communication between your external instance and AWS, see [Networking](#).

Issues running tasks on your external instance

If your tasks or containers fail to run on your external instance, the most common causes are either network or permission related. If your containers are pulling their images from Amazon ECR or are configured to send container logs to CloudWatch Logs, your task definition must specify a valid task execution IAM role. Without a valid task execution IAM role, your containers will fail to start. For more information about network related issues, see [External instance network issues](#).

Important

Amazon ECS provides the Amazon ECS logs collection tool. You can use it to collect logs from your external instances for troubleshooting purposes. For more information, see [Collecting container logs with Amazon ECS logs collector](#).

Troubleshoot Java class loading issues on Fargate

Java applications running on Fargate may encounter class loading issues after platform updates, particularly when the application relies on non-deterministic class loading behavior. This can manifest as dependency injection errors, Spring Boot failures, or other runtime exceptions that were not present in previous deployments.

Symptoms

You might experience the following symptoms:

- Spring Boot dependency injection errors
- `ClassNotFoundException` or `NoClassDefFoundError` exceptions

- Applications that previously worked on Fargate now fail intermittently
- The same container image works on Amazon EC2 but fails on Fargate
- Inconsistent behavior across deployments with identical container images

Causes

These issues typically occur due to:

- **Non-deterministic class loading:** Java applications that depend on the order in which classes are loaded from JAR files may fail when the underlying platform changes how files are accessed or cached.
- **Platform updates:** Fargate platform version updates may change the underlying file system behavior, affecting the order in which classes are discovered and loaded.
- **JAR file ordering dependencies:** Applications that implicitly rely on specific JAR loading sequences without explicit dependency management.

Resolution

To resolve Java class loading issues on Fargate, implement deterministic class loading practices:

Immediate fix

If you need an immediate workaround:

1. **Enforce JAR loading order:** Explicitly specify the order in which JAR files should be loaded in your application's classpath configuration.
2. **Use explicit dependency management:** Ensure all dependencies are explicitly declared in your build configuration (Maven, Gradle, etc.) rather than relying on transitive dependencies.

Long-term best practices

Implement these practices to prevent future class loading issues:

1. **Make class loading deterministic:**
 - Use explicit dependency declarations in your build files
 - Avoid relying on classpath scanning order

- Use dependency management tools to resolve version conflicts
- Use the Java Virtual Machine (JVM) options such as `-verbose:classpath` to get information about classes loaded by JVM.

2. Spring Boot applications:

- Use `@ComponentScan` with explicit base packages
- Avoid auto-configuration conflicts by explicitly configuring beans
- Use `@DependsOn` annotations to control bean initialization order

3. Build configuration:

- Use dependency management sections in Maven or Gradle
- Exclude transitive dependencies that cause conflicts
- Use tools like Maven Enforcer Plugin to detect dependency issues

4. Testing:

- Test your application with different JVM implementations
- Run integration tests that simulate different deployment environments
- Use tools to analyze classpath conflicts during development

Prevention

To prevent Java class loading issues in future deployments:

- **Follow deterministic class loading practices:** Design your application to not depend on the order in which classes are loaded from the classpath.
- **Use explicit dependency management:** Always explicitly declare all required dependencies and their versions in your build configuration.
- **Test across environments:** Regularly test your applications across different environments and platform versions to identify potential issues early.
- **Monitor platform updates:** Stay informed about Fargate platform updates and test your applications with new platform versions before they affect production workloads.

AWS Fargate throttling quotas

AWS Fargate limits Amazon ECS tasks and Amazon EKS pods launch rates to quotas (formerly referred to as limits) using a [token bucket algorithm](#) for each AWS account on a per-Region basis.

With this algorithm, your account has a bucket that holds a specific number of tokens. The number of tokens in the bucket represents your rate quota at any given second. Each customer account has a tasks and pods token bucket that depletes based on the number of tasks and pods launched by the customer account. This token bucket has a bucket maximum that allows you to periodically make a higher number of requests, and a refill rate that allows you to sustain a steady rate of requests for as long as needed.

For example, the tasks and pods token bucket size for a Fargate customer account is 100 tokens, and the refill rate is 20 tokens per second. Therefore, you can immediately launch up to 100 Amazon ECS tasks and Amazon EKS pods per customer account, with a sustained launch rate of 20 Amazon ECS tasks and Amazon EKS pods per second.

Actions	Bucket maximum capacity (or Burst rate)	Bucket refill rate (or Sustained rate)
Fargate Resource rate quota for On Demand Amazon ECS tasks and Amazon EKS pods ¹	100	20
Fargate Resource rate quota for Spot Amazon ECS tasks	100	20

¹Accounts launching only Amazon EKS pods have a burst rate of 20 with a sustained pod launch rate of 20 pod launches per second when using the platform versions called out in the [Amazon EKS platform versions](#).

Throttling the RunTask API in Fargate

In addition, Fargate limits the request rate when launching tasks using the Amazon ECS RunTask API using a separate quota. Fargate limits Amazon ECS RunTask API requests for each AWS account on a per-Region basis. Each request that you make removes one token from the bucket. We do this to help the performance of the service, and to ensure fair usage for all Fargate customers. API calls are subject to the request quotas whether they originate from the Amazon Elastic Container Service console, a command line tool, or a third-party application. The rate quota for calls to the Amazon ECS RunTask API is 20 calls per second (burst and sustained). Each call to this API can, however, launch up to 10 tasks. This means you can launch 100 tasks in one second by making 10 calls to this API, requesting 10 tasks to be launched in each call. Similarly, you could

also make 20 calls to this API, requesting 5 tasks to be launched in each call. For more information on API throttling for Amazon ECS RunTask API, see [API request throttling](#) in the Amazon ECS API Reference.

In practice, task and pod launch rates are also dependent on other considerations such as container images to be downloaded and unpacked, health checks and other integrations enabled, such as registering tasks or pods into a load balancer. Customers see variations in task and pod launch rates compared with the quotas represented earlier based on the features that customers enable.

Adjusting rate quotas in Fargate

You can request an increase for Fargate rate throttling quotas for your AWS account. For more information, see [Requesting a quota increase](#) in the *Service Quotas User Guide*.

Troubleshooting Amazon ECS Managed Instances

Use the following procedures to troubleshoot Amazon ECS Managed Instances, including common issues, diagnostic techniques, and resolution steps.

Prerequisites

Before troubleshooting Amazon ECS Managed Instances, ensure that you have the following requirements in place.

- The AWS CLI is installed and configured with appropriate permissions

For more information, see [Installing or updating to the latest version of the AWS Command Line Interface](#) in the *AWS Command Line Interface User Guide*.

- Access to a cluster with Amazon ECS Managed Instances capacity provider. For more information, see [the section called “Creating a cluster for Amazon ECS Managed Instances”](#).

Common troubleshooting scenarios

Viewing Amazon ECS Managed Instances container agent logs

You can view these Amazon ECS log files in Amazon ECS Managed Instances by connecting to a privileged container running in the instance.

Diagnostic steps

Deploy a debug container with privileges and Linux capabilities as an Amazon ECS task:

Set the following environment variables.

Replace the *user-input* with your values.

```
export ECS_CLUSTER_NAME="your-cluster-name"
export AWS_REGION="your-region"
export ACCOUNT_ID="your-account-id"
```

Create a task definition using a CLI JSON file called `node-debugger.json`.

```
cat << EOF > node-debugger.json
{
  "family": "node-debugger",
  "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
  "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
  "cpu": "256",
  "memory": "1024",
  "networkMode": "host",
  "pidMode": "host",
  "requiresCompatibilities": ["MANAGED_INSTANCES", "EC2"],
  "containerDefinitions": [
    {
      "name": "node-debugger",
      "image": "public.ecr.aws/amazonlinux/amazonlinux:2023",
      "essential": true,
      "privileged": true,
      "command": ["sleep", "infinity"],
      "healthCheck": {
        "command": ["CMD-SHELL", "echo debugger || exit 1"],
        "interval": 30,
        "retries": 3,
        "timeout": 5
      },
      "linuxParameters": {
        "initProcessEnabled": true
      },
      "mountPoints": [
        {
          "sourceVolume": "host-root",
```

```

        "containerPath": "/host",
        "readOnly": false
    }
],
"logConfiguration": {
    "logDriver": "awslogs",
    "options": {
        "awslogs-group": "/aws/ecs/node-debugger",
        "awslogs-create-group": "true",
        "awslogs-region": "${AWS_REGION}",
        "awslogs-stream-prefix": "ecs"
    }
}
}
],
"volumes": [
    {
        "name": "host-root",
        "host": {
            "sourcePath": "/"
        }
    }
]
}
EOF

```

Register, and then run the task. Run the following commands.

```

aws ecs register-task-definition --cli-input-json file://node-debugger.json

TASK_ARN=$(aws ecs run-task \
  --cluster $ECS_CLUSTER_NAME \
  --task-definition node-debugger \
  --enable-execute-command \
  --capacity-provider-strategy capacityProvider=managed-instances-default,weight=1 \
  --query 'tasks[0].taskArn' --output text)

# Wait for task to be running
aws ecs wait tasks-running --cluster $ECS_CLUSTER_NAME --tasks $TASK_ARN

```

Connect to the container. Run the following command.

```
aws ecs execute-command \
```

```
--cluster $ECS_CLUSTER_NAME \  
--task $TASK_ARN \  
--container node-debugger \  
--interactive \  
--command "/bin/sh"
```

Check the Amazon ECS agent logs:

In the interactive session of the container, run the following commands:

```
# Install required tools  
yum install -y util-linux-core  
  
# View ECS agent logs  
nsenter -t 1 -m -p cat /var/log/ecs/ecs-agent.log | tail -50  
  
# Check agent registration  
nsenter -t 1 -m -p grep "Registered container instance" /var/log/ecs/ecs-agent.log  
  
Example Output:  
  
{"level":"info","time":"2025-10-16T12:39:37.665","msg":"Registered container instance  
with cluster!"}  
  
# Verify capabilities  
nsenter -t 1 -m -p grep "Response contained expected value for attribute" /var/log/ecs/  
ecs-agent.log
```

Check agent metrics:

Run the following command to view the logs.

```
# View metrics logs  
nsenter -t 1 -m -p cat /var/log/ecs/metrics.log | tail -20
```

Task placement issues

The following are symptoms of task placement issues:

- Tasks stuck in PENDING state
- Tasks failing to start on Amazon ECS Managed Instances
- Insufficient resources errors

Diagnostic steps

Run the following commands to diagnose task placement issues and gather information about cluster capacity, container instances, and system services:

```
# Check cluster capacity
aws ecs describe-clusters --clusters cluster-name --include STATISTICS

# Check cluster capacity providers
aws ecs describe-clusters --clusters cluster-name --include STATISTICS --query
'clusters[].capacityProviders'

# List container instances
aws ecs list-container-instances --cluster cluster-name

# Check container instance details
aws ecs describe-container-instances --cluster cluster-name --container-
instances container-instance-arn

# Check container instance remaining resources CPU/Mem
aws ecs describe-container-instances --cluster $ECS_CLUSTER_NAME --container-
instances container-instance-arn --query 'containerInstances[].remainingResources'

# Check container instance Security Group
aws ecs describe-container-instances --cluster $ECS_CLUSTER_NAME --container-
instances container-instance-arn --query 'containerInstances[].ec2InstanceId' --output
text
aws ec2 describe-instances --instance-ids instance-id --query
'Reservations[0].Instances[0].SecurityGroups'
aws ec2 describe-security-groups --group-ids security-group-id
```

System service monitoring:

```
# Check Containerd status
nsenter -t 1 -m -p systemctl status containerd.service

# Check Amazon ECS container agent status
nsenter -t 1 -m -p systemctl status ecs
```

Resolution

To resolve task placement issues, follow these steps to ensure proper configuration and capacity:

- Verify task resource requirements vs available capacity
- Check placement constraints and strategies
- Ensure Amazon ECS Managed Instances capacity provider is configured
- Ensure that the task and container instance security group have an outbound rule that allow traffic for the Amazon ECS agent management endpoints

Networking issues

The following are symptoms of networking issues:

- Tasks unable to reach external services
- DNS resolution problems

Diagnostic steps

Network connectivity tests:

From the debug container, run the following commands:

Note

Confirm the security group attached to your capacity provider or Amazon ECS task is permitting the traffic.

```
# Install DNS Utility
yum install bind-utils -y

# Test DNS resolution
nslookup amazon.com

# Test external connectivity
curl -I https://amazon.com
```

Resource constraints

The following are symptoms of networking issues:

- Tasks killed due to memory limits

- CPU throttling
- Disk space issues

Diagnostic steps

Run commands to monitor the resources and container limits.

Resource monitoring:

```
# Check memory usage
nsenter -t 1 -m -p free -h

# Check disk usage
nsenter -t 1 -m -p lsblk

# Check disk usage
nsenter -t 1 -m -p df -h
```

Container Limits:

```
# Check OOM kills
nsenter -t 1 -m -p dmesg | grep -i "killed process"
```

Container instance agent disconnect issue

The following are symptoms of container instance agent disconnect issues:

- Container instances showing as disconnected in the Amazon ECS console
- Tasks failing to be placed on specific instances
- Agent registration failures in logs

Diagnostic steps

If there is an existing privilege task running on the host that ECS Exec can access, run the following commands to diagnose agent connectivity issues:

```
# check service status
nsenter -t 1 -m -p systemctl restart ecs
nsenter -t 1 -m -p systemctl restart containerd
```

```
# restart stopped services
nsenter -t 1 -m -p systemctl restart ecs
nsenter -t 1 -m -p systemctl restart containerd
```

Otherwise, force deregister the Amazon ECS Managed Instances. Run the following command:

```
# list ECS Managed Instance container
aws ecs list-container-instances --cluster managed-instances-cluster --query
'containerInstanceArns' --output text

# deregister the specific container instance
aws ecs deregister-container-instance \
  --cluster $ECS_CLUSTER_NAME \
  --container-instance container-instance-arn \
  --force
```

Resolution

To resolve agent disconnect issues, follow these steps:

- Verify IAM role permissions for the container instance
- Check security group rules allow outbound HTTPS traffic to ECS endpoints
- Ensure network connectivity to AWS services
- Restart the ECS agent service if necessary: `nsenter -t 1 -m -p systemctl restart ecs`
- Verify the `ECS_CLUSTER` configuration in `/etc/ecs/ecs.config` matches your cluster name

Log Analysis in Amazon ECS Managed Instances

System logs

Use the following commands to examine system logs and identify potential issues with the managed instance:

```
# Check system messages
nsenter -t 1 -m -p journalctl --no-pager -n 50

# Check kernel logs
nsenter -t 1 -m -p dmesg | tail -20

# Check for disk space errors
```

```
nsenter -t 1 -m -p journalctl --no-pager | grep -i "no space\|disk full\|enospc"
```

Use the EC2 AWS CLI to get the console output from a Amazon ECS Managed Instance

Use the Amazon EC2 instance ID to retrieve the console output.

Replace the *user-input* with your values.

```
aws ec2 get-console-output --instance-id instance-id --latest --output text
```

Cleanup

Run the following to stop the debug task and deregister the task definition.

```
# Stop debug task
aws ecs stop-task --cluster $ECS_CLUSTER_NAME --task $TASK_ARN

# Deregister task definition (optional)
aws ecs deregister-task-definition --task-definition node-debugger
```

Additional resources

For more information about troubleshooting Amazon ECS Managed Instances, see the following resources:

- [the section called “Amazon ECS Managed Instances errors”](#)
- [Troubleshooting](#)
- [the section called “Container agent configuration”](#)
- [the section called “Monitor Amazon ECS containers with ECS Exec”](#)

Troubleshooting Amazon ECS Managed Instances

When launching tasks with Amazon ECS Managed Instances, Amazon ECS first attempts to place tasks on existing capacity and requests additional capacity for tasks that cannot be placed. If instance provisioning fails, the Amazon EC2 request ID is included in the task failure message. You can use this request ID to look up details of the failed request in CloudTrail for further troubleshooting.

Note

If you choose to apply least-privilege permissions and specify your own permissions for the instance profile instead of using the `AmazonECSInstanceRolePolicyForManagedInstances` managed policy, you can add the following permissions to help with troubleshooting task-related issues with Amazon ECS Managed Instances:

- `ecs:StartTelemetrySession`
- `ecs:PutSystemLogEvents`

Task definition is incompatible with Amazon ECS Managed Instances

Common cause

This error occurs when your task definition contains parameters or configurations that are not supported by Amazon ECS Managed Instances. Common incompatibilities include unsupported network modes, task roles, or resource requirements.

Resolution

1. Verify that your task definition uses `requiresCompatibilities` set to `MANAGED_INSTANCES`.
2. Ensure your task definition uses the `awsvpc` network mode.
3. Check that CPU and memory values are within supported ranges for Amazon ECS Managed Instances.
4. Review the detailed error message for specific incompatibility details.

Capacity provider not associated with cluster

Common cause

This error occurs when the capacity provider specified in your capacity provider strategy is not associated with the cluster or does not exist.

Resolution

1. Verify that the capacity provider exists in your account and region.

2. Associate the capacity provider with your cluster using the Amazon ECS console or CLI.
3. Ensure the capacity provider is in ACTIVE status before using it.

Infrastructure role permission errors

Common cause

This error occurs when the Amazon ECS infrastructure role lacks the necessary permissions to perform Amazon EC2 operations on your behalf, or when the role cannot be assumed due to trust relationship issues.

Resolution

1. Verify that your infrastructure role has the proper trust relationship with Amazon ECS.
2. Ensure the role has the required Amazon EC2 permissions including `ec2:RunInstances`, `ec2:DescribeInstances`, and `iam:PassRole`.
3. Check the encoded authorization failure message in CloudTrail for specific permission details.
4. Update the role policy to include missing permissions identified in the error message.

VcpuLimitExceeded error

Common cause

This error occurs when you've reached your vCPU service quota for the instance type family in the current region. Amazon ECS Managed Instances cannot launch additional instances until capacity is available.

Resolution

1. Request a service quota increase for the affected instance type family through the AWS Support Center.
2. Consider using different instance types that fall under a different vCPU quota category.
3. Terminate unused Amazon EC2 instances to free up vCPU capacity.
4. Review your capacity provider configuration to use instance types with lower vCPU requirements.

InsufficientCapacity and related capacity errors

Common cause

These errors occur when AWS doesn't have sufficient capacity to fulfill your instance request. This can include insufficient instance capacity, address capacity, or volume capacity in the requested Availability Zone.

Resolution

1. Try launching instances in different Availability Zones by configuring multiple subnets in your capacity provider.
2. Consider using different instance types that may have more available capacity.
3. Wait and retry the operation as capacity availability changes frequently.
4. For persistent capacity needs, consider using Reserved Instances or Savings Plans.

UnauthorizedOperation error

Common cause

This error occurs when the Amazon ECS service doesn't have the necessary permissions to perform Amazon EC2 operations or pass IAM roles. Common scenarios include missing `ec2:RunInstances` permissions or `iam:PassRole` permissions for the instance profile.

Resolution

1. Verify that your Amazon ECS infrastructure role has the necessary permissions to launch Amazon EC2 instances.
2. Ensure the infrastructure role has `iam:PassRole` permissions for the instance profile used by your Amazon ECS Managed Instances.
3. Check the encoded authorization failure message in CloudTrail for specific permission details.
4. Update the role policy to include the missing permissions identified in the error message.

Task timed out waiting for capacity

Common cause

This error occurs when instances take longer than expected to launch and register with the cluster. This can happen due to Amazon EC2 capacity constraints, instance launch failures, or network connectivity issues.

Resolution

1. Check Amazon EC2 service health in your region for any ongoing issues.
2. Verify that your subnets have sufficient IP addresses available.
3. Ensure your security groups allow the necessary traffic for Amazon ECS agent communication.
4. Consider using multiple Availability Zones to improve capacity availability.
5. Retry the task launch operation as capacity constraints are often temporary.

Network configuration errors

Common cause

These errors occur when there are mismatches between your task's network requirements and the capacity provider's network configuration, such as VPC mismatches or missing network configuration.

Resolution

1. Verify that your capacity provider is configured with the correct VPC and subnets.
2. Ensure that security groups and subnets belong to the same VPC.
3. Check that your task definition's network configuration is compatible with the capacity provider.
4. Update your capacity provider configuration with the correct network settings.

Capacity provider can't be deleted due to stuck instances

Common cause

These errors occur when Amazon ECS Managed Instances are stuck in an ACTIVE or DRAINING state but there are no running tasks on the instances.

Resolution

To allow the deletion of the capacity provider to proceed, you can force deregister the instances that are stuck using the following command.

```
aws ecs deregister-container-instance \
  --cluster arn:aws:ecs:us-east-1:111122223333:cluster/MyCluster \
  --container-instance arn:aws:ecs:us-east-1:111122223333:container-instance/
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE \
  --force
```

Troubleshooting Amazon ECS Amazon ECS Managed Instances errors

The following are some Amazon ECS Managed Instances error messages and actions that you can take to fix the errors.

Amazon ECS Managed Instances Provider is not supported without a Cluster Name

This error occurs when you try to create a capacity provider with a Amazon ECS Managed Instances provider parameter but no cluster field.

To resolve this issue, specify a valid cluster name when creating the capacity provider.

The Amazon ECS Managed Instances provider details are required when creating a Amazon ECS Managed Instances capacity provider

This error appears when you try to create a capacity provider with Amazon ECS Managed Instances provider but do not supply the actual object.

To correct this issue, specify valid Amazon ECS Managed Instances provider details and try again.

Amazon ECS Managed Instances capacity provider must specify an instance profile

You'll see this error when you try to create a capacity provider with Amazon ECS Managed Instances provider but do not provide the `Ec2InstanceProfile`.

To resolve this, specify a valid EC2 instance profile for your Amazon ECS Managed Instances capacity provider. For more information, see [the section called "Amazon ECS Managed Instances instance profile"](#).

Amazon ECS Managed Instances capacity provider must specify an InfrastructureRole

This message appears when you try to create a capacity provider with Amazon ECS Managed Instances provider but do not provide the InfrastructureRole.

To correct this, specify a valid infrastructure role for your Amazon ECS Managed Instances capacity provider. For more information, see [the section called "Infrastructure IAM role"](#).

Amazon ECS Managed Instances capacity provider must specify a Network Configuration

You'll encounter this error when you try to create a capacity provider with Amazon ECS Managed Instances provider but do not provide network configuration.

To resolve this, specify a valid network configuration with non-empty subnets for your Amazon ECS Managed Instances capacity provider. For more information, see [the section called "Task networking for Amazon ECS Managed Instances"](#).

No instance types satisfy the instance requirements specified in the Amazon ECS Managed Instances capacity provider

This happens when you try to create a capacity provider with Amazon ECS Managed Instances provider but no EC2 instance type satisfies the instance requirements.

To address this, review and adjust your instance requirements to match available EC2 instance types.

The specified infrastructure role arn is invalid

This error occurs when the InfrastructureRole does not follow the specific ARN format.

Expected format: `arn:partition:iam::account-id:role/role-name`. Specify a valid role ARN and try again. For more information, see [the section called "Infrastructure IAM role"](#).

The specified instance profile role arn is invalid

This error occurs when the instance profile does not follow the specific ARN format.

Expected format: `arn:partition:iam::account-id:instance-profile/profile-name`. Specify a valid role ARN and try again. For more information, see [the section called "Amazon ECS Managed Instances instance profile"](#).

The specified security group id is invalid

This error occurs when the security group ID does not follow the specific format.

Expected format: `sg-xxxxxxx` or `sg-xxxxxxxxxxxxxxxxxxx` (8 or 17 characters including letters (lowercase), and numbers after 'sg-'). Specify a security group ID and try again.

The specified subnet id is invalid

This error occurs when the subnet ID does not follow the specific format.

Expected format: `subnet-xxxxxxx` or `subnet-xxxxxxxxxxxxxxxxxxx` (8 or 17 characters including letters (lowercase), and numbers after 'subnet-'). Specify a subnet ID and try again.

Amazon ECS Managed Instances capacity provider must specify a Network Configuration with non empty subnets

This error occurs when you try to create a capacity provider with Amazon ECS Managed Instances provider and network configuration with empty subnets.

To resolve this, specify a valid network configuration with non-empty subnets for your Amazon ECS Managed Instances capacity provider.

Amazon ECS Managed Instances capacity provider cannot have number of tags exceeding maximum allowed value

This error occurs when you try to create a capacity provider with more than the maximum allowed number of tags (45).

To resolve this, reduce the number of tags to 45 or fewer and try again.

Invalid value for propagateTags

This error occurs when you try to create a capacity provider with an invalid `propagateTags` value.

The value must be one of the valid `propagateTags` values. Specify a valid value and try again.

The specified capacity provider already exists with cluster scope

This error occurs when you try to create, update, or delete a Amazon ECS Managed Instances capacity provider without a cluster name, but there is an existing cluster-scoped capacity provider.

To change the configuration of or delete the capacity provider, update or delete the capacity provider with the cluster parameter.

The specified capacity provider already exists with account or a different cluster

This error occurs when you try to create, update, or delete a Amazon ECS Managed Instances capacity provider with a cluster field, while there is an existing account-scoped capacity provider or an existing cluster-scoped capacity provider for a different cluster.

To resolve this, use a different capacity provider name or work with the existing capacity provider.

Capacity provider active cluster name does not match the cluster name in the request

This error occurs when the capacity provider's active cluster name does not match the cluster name specified in the request.

Ensure that the cluster name in your request matches the capacity provider's associated cluster.

Capacity provider is not Amazon ECS Managed Instances capacity provider

This error occurs when you try to use a capacity provider that is not a Amazon ECS Managed Instances capacity provider in a context that requires one.

Ensure you are using a valid Amazon ECS Managed Instances capacity provider for your request.

CapacityProvider name must not be blank

This error occurs when you try to create a capacity provider without specifying a capacity provider name.

Specify a valid capacity provider name and try again.

A name is required when updating a capacity provider

This error occurs when you try to update a capacity provider without specifying a capacity provider name.

Specify a valid name and try again.

Tags can not be null or have null elements

This error occurs when you try to tag a capacity provider with null values.

Ensure all tag values are properly specified and not null.

Amazon ECS API failure reasons

When an API action that you have triggered through the Amazon ECS API, console, or the AWS CLI exits with a `failures` error message, the following might assist in troubleshooting the cause. The failure returns a reason and the Amazon Resource Name (ARN) of the resource associated with the failure.

Many resources are Region-specific, so when using the console ensure that you set the correct Region for your resources. When using the AWS CLI, make sure that your AWS CLI commands are being sent to the correct Region with the `--region region` parameter.

For more information about the structure of the `Failure` data type, see [Failure](#) in the *Amazon Elastic Container Service API Reference*.

The following are examples of failure messages that you might receive when running API commands.

API action	Failure reason or Stopped reason	Cause
<code>DescribeClusters</code>	MISSING	The specified cluster wasn't found. Verify the spelling of the cluster name.
<code>DescribeInstances</code>	MISSING	The specified container instance wasn't found. Verify that you specified the cluster the container instance is registered to and that both the container instance ARN or ID is correct.
<code>DescribeServices</code>	MISSING	The specified service wasn't found. Verify that the correct cluster or Region is specified

API action	Failure reason or Stopped reason	Cause
		and that the service ARN or name is valid.
DescribeTasks	MISSING	The specified task wasn't found. Verify the correct cluster or Region is specified and that both the task ARN or ID is valid.

API action	Failure reason or Stopped reason	Cause
DescribeTasks	TaskFailedToStart: RESOURCE:*	<p>For RESOURCE:CPU errors, the number of CPUs requested by the task are unavailable on your container instances. This generally happens when the CPU unit requirement in your task definition is larger than the CPU size of the Amazon EC2 instances defined in the Auto Scaling group mapped to the capacity provider. You need to check your capacity provider configuration.</p> <p>For RESOURCE:MEMORY errors, the amount of memory requested by the task are unavailable on your container instances. This generally happens when the memory amount requirement in your task definition is larger than the supported memory on the Amazon EC2 instances defined in the Auto Scaling group mapped to the capacity provider. You need to check your capacity provider configuration.</p>

API action	Failure reason or Stopped reason	Cause
	TaskFailedToStart: AGENT	<p>The container instance that you attempted to launch a task onto has an agent that's currently disconnected. To prevent extended wait times for task placement, the request was rejected.</p> <p>For information about how to troubleshoot an agent that's disconnected, see How do I troubleshoot a disconnected Amazon ECS agent.</p>
	TaskFailedToStart: MemberOf placement constraint unsatisfied	<p>There is no container instance that meets the placement constraints defined in your task definition.</p>

API action	Failure reason or Stopped reason	Cause
	TaskFailedToStart: ATTRIBUTE	Your task definition contains a parameter that requires a specific container instance attribute that isn't available on your container instances. For example, if your task uses the <code>awsvpc</code> network mode, but there are no instances in your specified subnets with the <code>ecs.capability.task-eni</code> attribute. For more information about which attributes are required for specific task definition parameters and agent configuration variables, see Amazon ECS task definition parameters for Fargate and Amazon ECS container agent configuration .
	TaskFailedToStart: NO ACTIVE INSTANCES	There are no active instances in your capacity provider. For information about how to manage your Auto Scaling groups, see Auto Scaling groups in the <i>Amazon EC2 Auto Scaling User Guide</i> .

API action	Failure reason or Stopped reason	Cause
GetTaskProtection	TaskFailedToStart: EMPTY_CAPACITY_PROVIDER	There are no instances in your cluster. This is most likely because of an empty capacity provider, or because the instances in the capacity provider are not registered to the cluster. For information about how to manage your Auto Scaling groups, see Auto Scaling groups in the <i>Amazon EC2 Auto Scaling User Guide</i> .
	MISSING	The specified task wasn't found. Verify that the cluster name or ARN and the task ARN or ID are valid.
	TASK_NOT_VALID	The specified task isn't part of an Amazon ECS service. Only Amazon ECS service-managed tasks can be protected. Verify the task ARN or ID and try again.

API action	Failure reason or Stopped reason	Cause
RunTask or StartTask	RESOURCE : *	<p>The resource or resources that are requested by the task are unavailable on the container instances in the cluster. If the resource is CPU, memory, ports, or elastic network interfaces, you might need to add additional container instances to your cluster.</p> <p>For RESOURCE : ENI errors, your cluster doesn't have any available elastic network interface attachment points, which are required for tasks that use the awsvpc network mode. Amazon EC2 instances have a limit to the number of network interfaces that can be attached to them, and the primary network interface counts as one. For more information about how many network interfaces are supported for each instance type, see IP Addresses Per Network Interface Per Instance Type in the <i>Amazon EC2 User Guide</i>.</p> <p>For RESOURCE : GPU errors, the number of GPUs requested by the task are unavailable and you might</p>

API action	Failure reason or Stopped reason	Cause
		need to add GPU-enabled container instances to your cluster. For more information, see Amazon ECS task definitions for GPU workloads .
	AGENT	<p>The container instance that you attempted to launch a task onto has an agent that's currently disconnected. To prevent extended wait times for task placement, the request was rejected.</p> <p>For information about how to troubleshoot an agent that's disconnected, see How do I troubleshoot a disconnected Amazon ECS agent.</p>
	LOCATION	The container instance that you attempted to launch a task onto is in a different Availability Zone than the subnets that you specified in your <code>awsVpcConfiguration</code> .

API action	Failure reason or Stopped reason	Cause
StartTask	ATTRIBUTE	Your task definition contains a parameter that requires a specific container instance attribute that isn't available on your container instances. For example, if your task uses the <code>awsvpc</code> network mode, but there are no instances in your specified subnets with the <code>ecs.capability.task-eni</code> attribute. For more information about which attributes are required for specific task definition parameters and agent configuration variables, see Amazon ECS task definition parameters for Fargate and Amazon ECS container agent configuration .
	MISSING	The container instance that you attempted to launch the task onto can't be found. Check if the wrong cluster or Region is specified, or the container instance ARN or ID is misspelled.
	INACTIVE	The container instance that you attempted to launch a task onto was previously deregistered with Amazon ECS and can't be used.

API action	Failure reason or Stopped reason	Cause
StopServiceDeployment	ECS deployment failed	A fraud account ran the StopServiceDeployment API.
TagResource	InvalidParameterException	The ARN for the service that you are tagging has the short format. You must migrate to the long format. For information about how to migrate the ARN, see Migrate an Amazon ECS short service ARN to a long ARN .
UpdateTaskProtection	DEPLOYMENT_BLOCKED	Can't set task protection as one or more protected tasks are preventing the service deployment from reaching a steady state. Unset task protection on existing tasks or wait until task protection expires.
	MISSING	The specified task wasn't found. Verify that the cluster name or ARN and the task ARN or ID are valid.
	TASK_NOT_VALID	The specified task isn't part of an Amazon ECS service. Only Amazon ECS service-managed tasks can be protected. Verify the task ARN or ID and try again.

 **Note**

Besides the failure scenarios described here, API operations can also fail due to exceptions, resulting in error responses. For a list of such exceptions, see [Common Errors](#).

Troubleshooting with Amazon Q Developer

You can use [Amazon Q Developer](#) in the Amazon ECS console to help diagnose and resolve issues with your Amazon ECS resources. For certain errors and status messages on containers, tasks, services, deployments, and task definitions, the console shows an **Inspect with Amazon Q Developer** option. When you choose this option, Amazon Q Developer analyzes the issue in context and suggests possible causes and remediation steps.

Required permissions

- Permissions to view the Amazon ECS resources you want to troubleshoot, such as clusters, services, tasks, and task definitions.
- [Permissions to use Amazon Q Developer](#) in the console.
- (Recommended) Permissions to view related logs and metrics, such as:
 - CloudWatch Logs
 - CloudWatch

Procedure

1. Open the console at <https://console.aws.amazon.com/ecs/v2>.
2. Determine the resource that you want to troubleshoot.

Resource	Steps
Tasks	<ol style="list-style-type: none">1. On the Clusters page, choose the cluster.2. Choose the Tasks tab.3. Choose the task that you want to investigate.

Resource	Steps
Containers	<ol style="list-style-type: none">1. On the Clusters page, choose the cluster.2. Choose the Tasks tab, and then choose the task.3. In the Containers section, choose the container you want to investigate.
Services and deployments	<ol style="list-style-type: none">1. On the Clusters page, choose the cluster.2. Choose the Services tab.3. Choose the service, and then review the Deployments section.
Task definitions	<ol style="list-style-type: none">1. In the navigation pane, choose Task definitions.2. Choose the task definition family, and then choose the revision that you want to investigate.

3. On the resource details page, locate the status or health reason that describes the issue.
4. Click the status reason to open the popover.
5. If available, choose **Inspect with Amazon Q Developer**.
6. Review the explanation and suggested remediation steps that Amazon Q Developer provides. Apply configuration or operational changes as appropriate for your environment.

Considerations

Consider the following when using Amazon Q Developer with Amazon ECS:

- **Button availability** - The "Inspect with Amazon Q Developer" button is only displayed for resources experiencing potential issues. This option is not available for healthy resources.
- **Read-only operations** - The Amazon Q Developer integration performs only read operations. It makes no mutating or write actions.
- **Cross-region processing** - Amazon Q Developer may process data across AWS regions to provide AI-powered analysis. For more information about cross-region processing, see [Cross-region processing in Amazon Q Developer](#).

- **Console only** - This integration is available only through the Console. It is not available through the AWS CLI, AWS APIs, or infrastructure as code tools.

Learn more

For more information about using Amazon Q Developer, see [Chatting with Amazon Q Developer about AWS](#).