ALIBABA CLOUD

Alibaba Cloud

Elastic Container Instance

Container

Document Version: 20220421

Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	**Danger:** Resetting will result in the loss of user configuration data.
Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	**Warning:** Restarting will cause business interruption. About 10 minutes are required to restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	**Notice:** If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	**Note:** You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.Configure startup commands and arguments for a container

Elastic Container Instance starts a container by using predefined arguments in the container image. If you did not configure startup commands or arguments when you created an image or if you want to modify them, you can configure the commands or arguments when you create an elastic container instance. This topic describes how to configure the commands and arguments that are run after a container is started.

Background information

If you want to override the default startup settings of an image, such as the working directory and startup commands and arguments, you can set the following fields in the configuration file of an elastic container instance:

- WorkingDir

  When you create an image, you can set the WORKDIR field to specify the working directory of a container. The startup commands of the container are run in the specified directory. For more information, see WORKDIR.

  You can override the WORKDIR value provided by the image by setting the WorkingDir field in the configuration file of an elastic container instance.

  > Note
  > - If the WORKDIR field is not specified in the image and the WorkingDir field is not specified in the elastic container instance, the working directory is set to the root directory.
  > - If the specified working directory does not exist, the system creates the directory.

- Command and Arg

  When you create an image, you can set the CMD and ENTRYPOINT fields to specify the commands and arguments that are run after the container is started. For more information, see ENTRYPOINT and CMD.

  You can override the CMD and ENTRYPOINT values provided by the image by setting the Command and Arg fields in the configuration file of an elastic container instance. The following table describes the validity rules.

ENTRYPOINT value	CMD value	Command value	Arg value	Command that is run	Description
[mkdir]	[/data/backup]	Not specified	Not specified	[mkdir /data/backup]	Neither of Command and Arg is specified. The commands and arguments specified by CMD and ENTRYPOINT are run.

ENTRYPOINT value	CMD value	Command value	Arg value	Command that is run	Description
[mkdir]	[/data/backup]	[cd]	Not specified	[cd]	Command is specified but Arg is not. Only commands specified by Command is run, and ENTRYPOINT and CMD are ignored.
[mkdir]	[/data/backup]	Not specified	[/opt/backup]	[mkdir /opt/backup]	Arg is specified but Command is not. The commands and arguments specified by Arg and ENTRYPOINT are run.
[mkdir]	[/data/backup]	[cd]	[/opt/backup]	[cd /opt/backup]	Both Command and Arg are specified, and the commands and arguments specified by Command and Arg are run.

Notice

The values of Command and Arg must be supported by the container image. Otherwise, the container fails to start.

Kubernetes mode

When you use the Kubernetes mode to create an elastic container instance, you can use the workingDir field to specify the working directory. You can also use the command and args fields to specify the startup commands and arguments. Example:

```
apiVersion: v1
kind: Pod
metadata:
  name: command-demo
spec:
  containers:
  - name: busybox
    image: busybox
    workingDir: /work
    command: ["printenv"]
    args: ["HOSTNAME", "KUBERNETES_PORT"]
  restartPolicy: OnFailure
```

For more information, see Define a Command and Arguments for a Container.

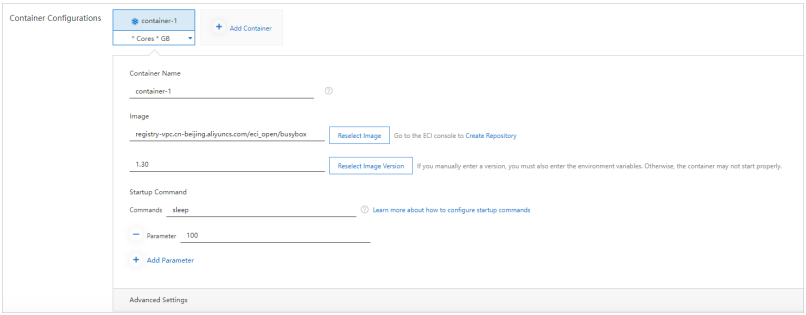
API mode

When you call the CreateContainerGroup operation to create an elastic container instance, you can use the WorkingDir parameter to specify the working directory. You can also use the Command and Arg parameters to specify the startup commands and arguments. The following table describes the parameters for the CreateContainerGroup operation. For more information, see CreateContainerGroup.

Parameter	Type	Required	Example	Description
Container.N.WorkingDir	String	No	/usr/local/	The working directory of the container.
Container.N.Command.N	RepeatList	No	sleep	The startup commands of the container. You can specify up to 20 commands.
Container.N.Arg.N	RepeatList	No	100	The arguments corresponding to the startup commands of the container. You can specify up to 10 arguments.

Console mode

When you create an elastic container instance on the Elastic Container Instance buy page, you can configure startup commands and arguments for containers by setting parameters in the **Container Configurations** section, as shown in the following figure.



2.Use probes to perform health checks on containers

This topic describes how to configure liveness probes and readiness probes to perform health checks on containers.

Background information

In Kubernetes, the kubelet uses liveness and readiness probes to periodically check the status and running conditions of containers.

- Liveness probe

  Liveness probes are used to check whether a container is working normally. If the check succeeds, the container is working normally. If the check fails, the system determines whether to restart the container based on the configured container restart policy. By default, if a liveness probe is not configured, the container is considered to be working normally all the time.

  Liveness probes can be applied in the following scenarios:

  - When an application is running but no further operations can be performed on it, liveness probes can capture the deadlock. The system then restarts the container to make the application run normally despite bugs.
  - Many applications running for a long period of time may eventually transition to the broken state, and cannot recover except by being restarted. You can use liveness probes to detect and remedy such situations.

- Readiness probe

  Readiness probes are used to check whether a container is ready to serve requests. If the check succeeds, the container is ready to receive business requests. If the check fails, the container is not ready and the system stops sending requests to the container until the recheck succeeds.

  Readiness probes can be applied in the following scenarios:

  Applications are temporarily unable to serve external traffic. For example, an application may need to load a large amount of data or configuration files during startup. In this case, if you do not want to terminate the application or send it requests, you can use readiness probes to detect and mitigate such situations.

Kubernetes mode

When you create an elastic container instance by using Kubernetes, you can configure liveness probes and readiness probes by using the livenessProbe and readinessProbe fields of the container. Examples:

- Configure a liveness probe

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: liveness-exec
spec:
  containers:
  - name: liveness
    image: busybox:latest
    args:
    - /bin/sh
    - -c
    - touch /tmp/healthy; sleep 30; rm -rf /tmp/healthy; sleep 600
    # Configure a liveness probe to check containers by using the command line.
    livenessProbe:
      exec:
        command:
        - cat
        - /tmp/healthy
      initialDelaySeconds: 5    #Start to check a container 5 seconds after the container
is started.
      periodSeconds: 5    #Perform a check every 5 seconds.
```

- Configure a readiness probe

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: readiness
  name: readiness-exec
spec:
  containers:
  - name: readiness
    image: busybox:latest
    args:
    - /bin/sh
    - -c
    - touch /tmp/healthy; sleep 30; rm -rf /tmp/healthy; sleep 600
    # Configure a readiness probe to check containers by using the command line.
    readinessProbe:
      exec:
        command:
        - cat
        - /tmp/healthy
      initialDelaySeconds: 5    #Start to check a container 5 seconds after the container
is started.
      periodSeconds: 5    #Perform a check every 5 seconds.
```

For more information, see Configure Liveness, Readiness and Startup Probes.

API mode

When you create an elastic container instance by calling the CreateContainerGroup operation, you can use the LivenessProbe and ReadinessProbe parameters to configure liveness probes and readiness probes. The following table describes the parameters. For more information, see CreateContainerGroup.

- LivenessProbe related parameters

Parameter	Type	Required	Example	Description
Container.N.LivenessProbe.HttpGet.Path	String	No	/healthyz	The path to which HTTP GET requests are sent for health checks.
Container.N.LivenessProbe.HttpGet.Port	Integer	No	8888	The port to which HTTP GET requests are sent for health checks.
Container.N.LivenessProbe.HttpGet.Scheme	String	No	HTTP	The protocol type of HTTP GET requests when you use HTTP requests for health checks. Valid values: • HTTP • HTTPS
Container.N.LivenessProbe.InitialDelaySeconds	Integer	No	5	The number of seconds after the container is started before a check is started.
Container.N.LivenessProbe.PeriodSeconds	Integer	No	1	The interval for checking the container. Default value: 10. Minimum value: 1. Unit: seconds.
Container.N.LivenessProbe.SuccessThreshold	Integer	No	1	The minimum number of consecutive times that a health check must succeed after a failed check before the check is declared successful. Default value: 1. Set the value to 1.

Parameter	Type	Required	Example	Description
Container.N.LivenessProbe.FailureThreshold	Integer	No	3	The minimum number of consecutive times that a health check must fail after a successful check before the check is declared failed. Default value: 3.
Container.N.LivenessProbe.TimeoutSeconds	Integer	No	1	The timeout period of the check. Default value: 1. Minimum value: 1. Unit: seconds.
Container.N.LivenessProbe.Exec.Command.N	RepeatList	No	cat /tmp/healthy	Command N to be executed in the container when the health check is performed by using the command line.
Container.N.LivenessProbe.TcpSocket.Port	Integer	No	8000	The port detected by Transmission Control Protocol (TCP) socket when health checks are performed by using TCP socket.

- ReadinessProbe related parameters

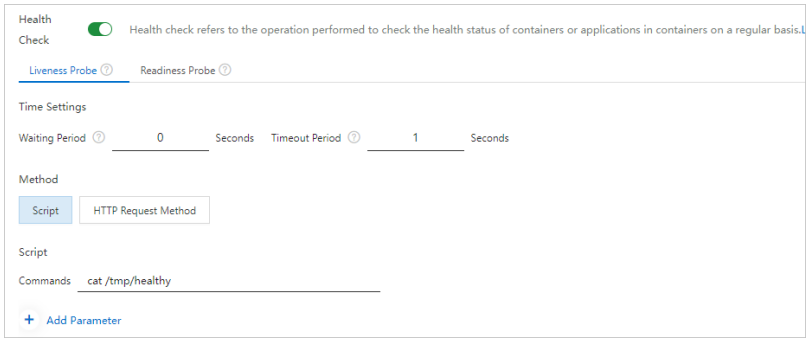
Parameter	Type	Required	Example	Description
Container.N.ReadinessProbe.HttpGet.Path	String	No	/healthyz	The path to which HTTP GET requests are sent for health checks.
Container.N.ReadinessProbe.HttpGet.Port	Integer	No	8888	The port to which HTTP GET requests are sent for health checks.
Container.N.ReadinessProbe.HttpGet.Scheme	String	No	HTTP	The protocol type of HTTP GET requests when you use HTTP requests for health checks. Valid values: • HTTP • HTTPS

Parameter	Type	Required	Example	Description
Container.N.ReadinessProbe.InitialDelaySeconds	Integer	No	5	The number of seconds after the container is started before probes are started.
Container.N.ReadinessProbe.PeriodSeconds	Integer	No	1	The interval for checking the container. Default value: 10. Minimum value: 1. Unit: seconds.
Container.N.ReadinessProbe.SuccessThreshold	Integer	No	1	The minimum number of consecutive times that a health check must succeed after a failed check before the check is declared successful. Default value: 1. Set the value to 1.
Container.N.ReadinessProbe.FailureThreshold	Integer	No	3	The minimum number of consecutive times that a health check must fail after a successful check before the check is declared failed. Default value: 3.
Container.N.ReadinessProbe.TimeoutSeconds	Integer	No	1	The timeout period of the check. Default value: 1. Minimum value: 1. Unit: seconds.
Container.N.ReadinessProbe.Exec.Command.N	RepeatList	No	cat /tmp/healthy	Command N to be executed in the container when the health check is performed by using the command line.
Container.N.ReadinessProbe.TcpSocket.Port	Integer	No	8000	The port detected by TCP socket when health checks are performed by using TCP socket.

Console mode

When you create an elastic container instance in the Elastic Container Instance console, you can click **Advanced Settings** in the **Container Configurations** section and enable Health Check. The following figure shows the configuration details.

> **Note**
>
> When you configure health check in the Elastic Container Instance console, only the command line and HTTP request methods are supported. The TCP socket method is not supported.



The following table describes the parameters.

Parameter	Description
Time Settings	You must set Waiting Period and Timeout Period. • Waiting Period: The number of seconds to wait before the probe performs the first health check. • Timeout Period: The timeout period of the check. If a timeout error occurs, the check failed.
Method	Valid values: • Script: The probe runs commands in the container and checks the exit code of the command. If the exit code is 0, the check is successful. • HTTP Request Method: The probe sends an HTTP request to the container. If the returned status code is greater than or equal to 200 and less than 400, the check is successful.
Script	When you select Script for Method, you must configure the command line script to be executed in the container.
HTTP Request Method	When you select HTTP Request Method for Method, you must configure the path, port, and protocol for HTTP GET requests.

Configuration examples

In this example, an elastic container instance that has a liveness probe and readiness probe configured is created and contains an NGINX container. Then, service exceptions are simulated to check whether the configured probes take effect.

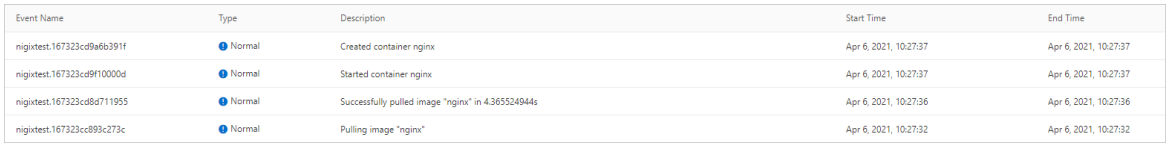
1. Create an elastic container instance by using SDK for Java.

Create an elastic container instance by using an NGINX image. The following sample code shows how to configure a liveness probe and readiness probe:

```
// Configure a liveness probe. After the container is running for 5 seconds, the kubelet runs the liveness probe on port 80 every 3 seconds. The timeout period of each probe is set to 10 seconds. The number of consecutive successes for a probe to be considered successful is set to 3, and the number of consecutive failures for a probe to considered failed is also set to 3.
CreateContainerGroupRequest.Container.ContainerProbe livenessProbe = new CreateContainerGroupRequest.Container.ContainerProbe();
livenessProbe.setTcpSocketPort(80);
livenessProbe.setInitialDelaySeconds(5);
livenessProbe.setPeriodSeconds(3);
livenessProbe.setFailureThreshold(3);
livenessProbe.setSuccessThreshold(1);
livenessProbe.setTimeoutSeconds(10);
// Configure a readiness probe. After the container is running for 5 seconds, the kubelet runs the readiness probe on port 80 every 3 seconds. The timeout period of each check is set to 10 seconds. The number of consecutive times that a health check must succeed before the check declared successful is set to 1, and the number of consecutive times that a probe must fail before the check is declared failed is set to 3.
CreateContainerGroupRequest.Container.ContainerProbe readinessProbe = new CreateContainerGroupRequest.Container.ContainerProbe();
readinessProbe.setTcpSocketPort(80);
readinessProbe.setInitialDelaySeconds(5);
readinessProbe.setPeriodSeconds(3);
readinessProbe.setFailureThreshold(3);
readinessProbe.setSuccessThreshold(3);
readinessProbe.setTimeoutSeconds(10);
```

2. View the related events after an elastic container instance is created.

   After an elastic container instance is created, view events about the instance and you can see that the instance starts normally.



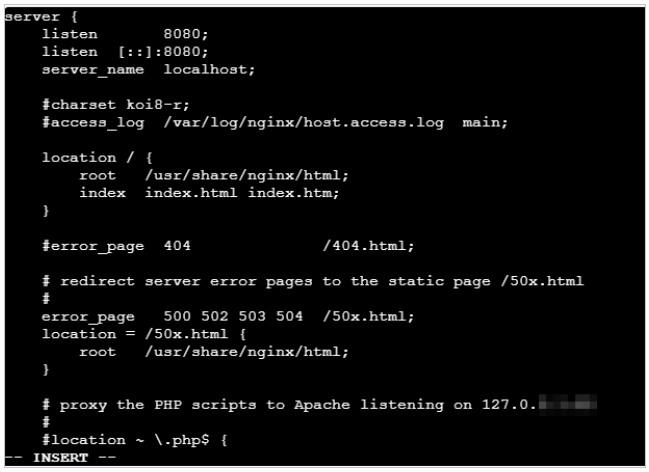
Event Name	Type	Description	Start Time	End Time
nigixtest.167323cd9a6b391f	Normal	Created container nginx	Apr 6, 2021, 10:27:37	Apr 6, 2021, 10:27:37
nigixtest.167323cd9f10000d	Normal	Started container nginx	Apr 6, 2021, 10:27:37	Apr 6, 2021, 10:27:37
nigixtest.167323cd8d711955	Normal	Successfully pulled image "nginx" in 4.365524944s	Apr 6, 2021, 10:27:36	Apr 6, 2021, 10:27:36
nigixtest.167323cc893c273c	Normal	Pulling image "nginx"	Apr 6, 2021, 10:27:32	Apr 6, 2021, 10:27:32

3. Change the NGINX listening port in the configuration file to simulate a service exception.

i. Change the NGINX listening port.

```
vi /etc/nginx/conf.d/default.conf
```

Example:



ii. Restart NGINX.

```
nginx -s reload
```

4. View the effective status of the probe.

A few seconds after you restart NGINX, the container automatically restarts. View events about the instance, and you can see that the container is restarted after the liveness probe and readiness probe each have three consecutive failures.

Event Name	Type	Description	Start Time	End Time
nigixtest.167325b909548ca9	Normal	Container nginx failed liveness probe, will be restarted	Apr 6, 2021, 11:02:47	Apr 6, 2021, 11:02:47
nigixtest.167325b90de4acb1	Normal	Container image "nginx" already present on machine	Apr 6, 2021, 11:02:47	Apr 6, 2021, 11:02:47
nigixtest.167325b90f476404	Normal	Created container nginx	Apr 6, 2021, 11:02:47	Apr 6, 2021, 11:02:47
nigixtest.167325b9138d313e	Normal	Started container nginx	Apr 6, 2021, 11:02:47	Apr 6, 2021, 11:02:47
nigixtest.167325b8f5c3a91a	Warning	Liveness probe failed: dial tcp 192.168.11.23:80: connect: connection refused	Apr 6, 2021, 11:02:47	Apr 6, 2021, 11:02:47
nigixtest.167325b8fb78342c	Warning	Readiness probe failed: dial tcp 192.168.11.23:80: connect: connection refused	Apr 6, 2021, 11:02:47	Apr 6, 2021, 11:02:47
nigixtest.167325b842f30264	Warning	Liveness probe failed: dial tcp 192.168.11.23:80: connect: connection refused	Apr 6, 2021, 11:02:44	Apr 6, 2021, 11:02:44
nigixtest.167325b848a667fb	Warning	Readiness probe failed: dial tcp 192.168.11.23:80: connect: connection refused	Apr 6, 2021, 11:02:44	Apr 6, 2021, 11:02:44
nigixtest.167325b790231587	Warning	Liveness probe failed: dial tcp 192.168.11.23:80: connect: connection refused	Apr 6, 2021, 11:02:41	Apr 6, 2021, 11:02:41
nigixtest.167325b795cfc219	Warning	Readiness probe failed: dial tcp 192.168.11.23:80: connect: connection refused	Apr 6, 2021, 11:02:41	Apr 6, 2021, 11:02:41

3.Obtain the metadata of a container

This topic describes how to obtain the metadata of a container.

Elastic Container Instance provides the following methods to expose pod (elastic container instance) information and container metadata to containers in the running state:

- Method 1: Use MetaServer to access metadata
- Method 2: Configure environment variables for a container
- Method 3: Use the Downward API

Method 1: Use MetaServer to access metadata

You can perform the following steps to obtain the metadata of an elastic container instance:

1. Connect to a container. For more information, see Debug an elastic container instance.
2. Run the following command to access the root directory of the metadata:

   ```
   curl http://100.100.100.200/latest/meta-data/
   ```

3. Append the name of a metadata item to the command to obtain information about the item.

   For example, run the following command to obtain the ID of the elastic container instance:

   ```
   curl http://100.100.100.200/latest/meta-data/instance-id
   ```

   The following table describes the metadata items that you can obtain for an elastic container instance.

Metadata item	Description
/dns-conf/nameservers	The Domain Name System (DNS) configurations of the elastic container instance.
/eipv4	The elastic IPv4 address of the elastic container instance.
/hostname	The hostname of the elastic container instance, which is the ContainerGroupName value.
/instance-id	The ID of the elastic container instance.
/mac	The media access control (MAC) address of the elastic container instance.

Metadata item	Description
/network/interfaces/	The MAC addresses of the network interface controllers (NICs).
/network/interfaces/macs/[mac]/network-interface-id	The ID of the NIC. Replace [mac] with the MAC address of the elastic container instance.
/network/interfaces/macs/[mac]/netmask	The subnet mask of the NIC.
/network/interfaces/macs/[mac]/vswitch-cidr-block	The IPv4 CIDR block of the vSwitch to which the NIC belongs.
/network/interfaces/macs/[mac]/vpc-cidr-block	The IPv4 CIDR block of the virtual private cloud (VPC) to which the NIC belongs.
/network/interfaces/macs/[mac]/private-ipv4s	The private IPv4 addresses assigned to the NIC.
/network/interfaces/macs/[mac]/vpc-ipv6-cidr-blocks	The IPv6 CIDR block of the VPC to which the NIC belongs. This item is applicable only to the elastic container instances that reside within VPCs and are assigned IPv6 addresses.
/network/interfaces/macs/[mac]/vswitch-id	The ID of the vSwitch in the same VPC as the security group of the NIC.
/network/interfaces/macs/[mac]/vpc-id	The ID of the VPC to which the security group of the NIC belongs.
/network/interfaces/macs/[mac]/primary-ip-address	The primary private IP address of the NIC.
/network/interfaces/macs/[mac]/gateway	The IPv4 gateway address of the VPC to which the NIC belongs.
/instance/max-netbw-egress	The maximum outbound internal bandwidth of the elastic container instance. Unit: Kbit/s.
/instance/max-netbw-ingerss	The maximum inbound internal bandwidth of the elastic container instance. Unit: Kbit/s.

Metadata item	Description
/network/interfaces/macs/[mac]/ipv6s	The IPv6 addresses assigned to the NIC. This item is applicable only to the elastic container instances that reside within VPCs and are assigned IPv6 addresses.
/network/interfaces/macs/[mac]/ipv6-gateway	The IPv6 gateway address of the VPC to which the NIC belongs.
/network/interfaces/macs/[mac]/vswitch-ipv6-cidr-block	The IPv6 CIDR block of the vSwitch to which the NIC is connected. This item is applicable only to the elastic container instances that reside within VPCs and are assigned IPv6 addresses.
/private-ipv4	The private IPv4 address of the elastic container instance.
/ntp-conf/ntp-servers	The address of the Network Time Protocol (NTP) server.
/owner-account-id	The ID of the Alibaba Cloud account to which the elastic container instance belongs.
/region-id	The region ID of the elastic container instance.
/serial-number	The serial number of the elastic container instance.
/vpc-id	The ID of the VPC to which the elastic container instance belongs.
/vpc-cidr-block	The CIDR block of the VPC to which the elastic container instance belongs.
/vswitch-cidr-block	The CIDR block of the vSwitch to which the elastic container instance is connected.
/vswitch-id	The ID of the vSwitch to which the elastic container instance is connected.

Metadata item	Description
/zone-id	The zone ID of the elastic container instance.
/ram/security-credentials/[role-name]	The temporary Security Token Service (STS) credentials generated for the Resource Access Management (RAM) role of the elastic container instance. You can obtain the STS credentials only after you specify a RAM role to an elastic container instance. Replace [role-name] with the name of the RAM role. If [role-name] is not specified, the name of the RAM role is returned.

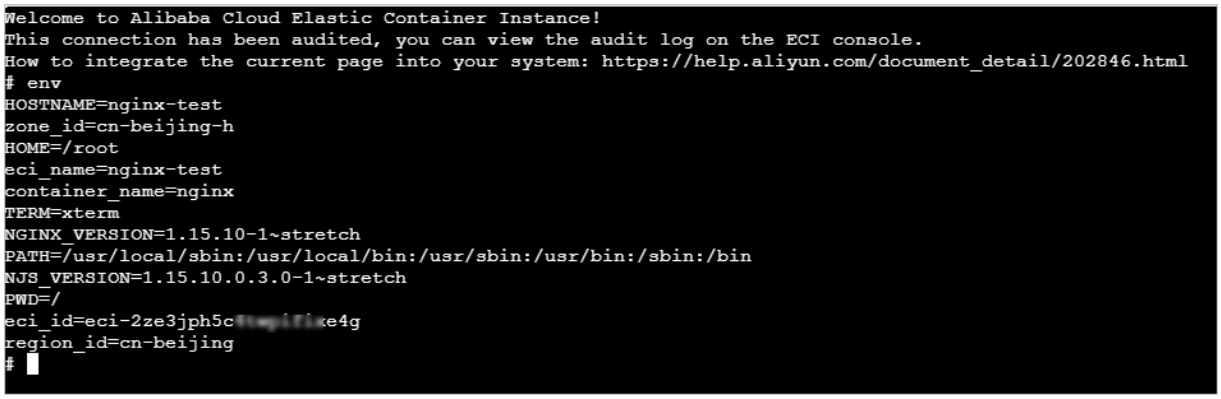
Method 2: Configure environment variables for a container

You can obtain the information about an elastic container instance by configuring the values of environment variables for a container in the instance. The metadata items of an elastic container instance that can be obtained in this manner include the instance ID, instance name, region ID of the instance, zone ID of the instance, and container name.

Key	Value	Description
eci_id	__ECI_ID__	The ID of the elastic container instance.
eci_name	__ECI_NAME__	The name of the elastic container instance.
region_id	__REGION_ID__	The region ID of the elastic container instance.
zone_id	__ZONE_ID__	The zone ID of the elastic container instance.
container_name	__CONTAINER_NAME__	The name of the container in the elastic container instance.

```
params = {
        'Container.1.Image': 'registry-vpc.cn-shanghai.aliyuncs.com/eci_open/nginx:alpine',
        'Container.1.Name': 'nginx',
        'SecurityGroupId': 'sg-uf6biempwqvodk7a****',
        'VSwitchId': 'vsw-uf6mhqg2wiq9iifhn****',
        'ContainerGroupName': 'test-env',
        # Configure environment variables.
        'Container.1.EnvironmentVar.1.Key': 'eci_id',
        'Container.1.EnvironmentVar.2.Key': 'eci_name',
        'Container.1.EnvironmentVar.3.Key': 'region_id',
        'Container.1.EnvironmentVar.4.Key': 'zone_id',
        'Container.1.EnvironmentVar.5.Key': 'container_name',
        'Container.1.EnvironmentVar.1.Value': '__ECI_ID__',
        'Container.1.EnvironmentVar.2.Value': '__ECI_NAME__',
        'Container.1.EnvironmentVar.3.Value': '__REGION_ID__',
        'Container.1.EnvironmentVar.4.Value': '__ZONE_ID__',
        'Container.1.EnvironmentVar.5.Value': '__CONTAINER_NAME__',
    }
```

You can log on to the Elastic Container Instance console and connect to the container to check whether the configured environment variables have taken effect. For more information, see Debug an elastic container instance.



Method 3: Use the Downward API

The Kubernetes Downward API provides the following methods to expose pod information to running containers:

- Pass pod information to environment variables of a container

  You can pass each piece of pod information as the value of a single environment variable to a container.

- Mount pod information as a file to the directory where a volume is mounted

  You can generate a file from pod information and mount the file to the directory where a volume is mounted in a container.

Alibaba Cloud Container Service for Kubernetes (ACK), Serverless Kubernetes (ASK), and Elastic Container Instance support the majority of fields commonly used by the Downward API.

- **Pass pod information to environment variables of a container**

You can use the Downward API to pass information such as the name, namespace, and IP address of a pod to environment variables of a container. The following table describes the pod parameters whose values can be passed to environment variables of a container.

Parameter	Description
metadata.name	The name of the pod.
metadata.namespace	The namespace of the pod.
metadata.uid	The UID of the pod.
metadata.labels['<KEY>']	The label value of the pod.
metadata.annotations['<KEY>']	The annotation value of the pod.
spec.serviceAccountName	The name of the pod service account.
spec.nodeName	The name of the node.
status.podIP	The IP address of the node.

The following Deployment sample code provides an example on how to pass the information of a pod to the environment variables of a container:

```
apiVersion: apps/v1 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
  name: vk-downward-env
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
        annotations:
            regionId: cn-beijing
            platform: Aliyun ECI
        labels:
            app: nginx
            env: test
    spec:
      containers:
```

```
  containers:
  - name: nginx
    image: nginx
    env:
    - name: MY_metadata.name
      valueFrom:
        fieldRef:
          fieldPath: metadata.name
    - name: MY_metadata.namespace
      valueFrom:
        fieldRef:
          fieldPath: metadata.namespace
    - name: MY_metadata.uid
      valueFrom:
        fieldRef:
          fieldPath: metadata.uid
    - name: MY_metadata.labels
      valueFrom:
        fieldRef:
          fieldPath: metadata.labels['env']
    - name: MY_metadata.annotations
      valueFrom:
        fieldRef:
          fieldPath: metadata.annotations['regionId']
    - name: MY_status.podIP
      valueFrom:
        fieldRef:
          fieldPath: status.podIP
    - name: MY_spec.serviceAccountName
      valueFrom:
        fieldRef:
          fieldPath: spec.serviceAccountName
    - name: MY_spec.nodeName
      valueFrom:
        fieldRef:
          fieldPath: spec.nodeName
```

Log on to the container and view the environment variables. You can find that fieldRef has taken effect. Examples of container environment variables to which pod information is passed after fieldRef takes effect:

```
root@default-vk-downward-env:/# env
MY_spec.nodeName=virtual-kubelet
MY_spec.serviceAccountName=default
MY_metadata.annotations=cn-beijing
MY_metadata.namespace=default
MY_metadata.uid=f4881309-f3dd-11e9-bcf9-9efaf54dcfa7
MY_metadata.name=vk-downward-env
MY_metadata.labels=test
MY_status.podIP=192.168.6.245
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_SERVICE_PORT=6443
PWD=/
PKG_RELEASE=1~buster
HOME=/root
KUBERNETES_PORT_443_TCP=tcp://172.22.*.*:443
NJS_VERSION=0.3.5
TERM=xterm
SHLVL=1
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_ADDR=172.22.*.*
KUBERNETES_SERVICE_HOST=192.168.*.*
KUBERNETES_PORT=tcp://172.22.*.*:443
KUBERNETES_PORT_443_TCP_PORT=443
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
NGINX_VERSION=1.17.4
_=/usr/bin/env
```

- **Volume files**

You can use the Downward API to mount pod information such as labels and annotations as a file to the directory where a volume is mounted in a container. The following table describes the pod parameters whose values that can be mounted to the directory where a volume is mounted in a container.

Parameter	Description
metadata.name	The name of the pod.
metadata.namespace	The namespace of the pod.
metadata.uid	The UID of the pod.
metadata.labels['<KEY>']	The label value of the pod.
metadata.annotations['<KEY>']	The annotation value of the pod.

Parameter	Description
metadata.labels	All labels of the pod.
metadata.annotations	All annotations of the pod.

> **Note**
>
> You can use the Downward API to mount pod fields, but you cannot mount container fields such as limits.cpu, requests.cpu, limits.memory, requests.memory, limits.ephemeral-storage, and requests.ephemeral-storage.

The following Deployment sample code provides an example on how to mount the information of a pod as a file to the directory where a volume is mounted in a container:

```
apiVersion: apps/v1beta2 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
  name: vk-downward-down-volume
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
        annotations:
            regionId: cn-beijing
            platform: Aliyun ECI
        labels:
            app: nginx
            env: test
    spec:
      containers:
      - name: nginx
        image: nginx
        volumeMounts:
        - name: podinfo
          mountPath: /etc/podinfo
          readOnly: false
      volumes:
      - name: podinfo
        downwardAPI:
          items:
            - path: "metadata.name"
              fieldRef:
                fieldPath: metadata.name
            - path: "metadata.namespace"
              fieldRef:
                fieldPath: metadata.namespace
            - path: "metadata.uid"
              fieldRef:
                fieldPath: metadata.uid
            - path: "metadata.labels"
              fieldRef:
                fieldPath: metadata.labels
            - path: "metadata.annotations"
              fieldRef:
                fieldPath: metadata.annotations
      nodeName: virtual-kubelet
```

Log on to the container and go to the directory where the volume is mounted. You can find that fieldRef has taken effect and that the pod information specified in fieldRef is stored in the directory. Example of pod information stored in the directory where a volume is mounted in a container:

```
Welcome to Alibaba Cloud Elastic Container Instance!
root@default-vk-downward-down-volume:/# cd /etc/podinfo/
root@default-vk-downward-down-volume:/etc/podinfo# ls
metadata.annotations  metadata.labels  metadata.name  metadata.namespace  metadata.uid
root@default-vk-downward-down-volume:/etc/podinfo# cat metadata.namespace
default
root@default-vk-downward-down-volume:/etc/podinfo# cat metadata.name
vk-downward-down-volume
root@default-vk-downward-down-volume:/etc/podinfo# cat metadata.uid
fa50b2b2-f3e3-11e9-bcf9-9efaf54dcfa7
root@default-vk-downward-down-volume:/etc/podinfo# cat metadata.annotations
platform="Aliyun ECI"
regionId="cn-beijing"
root@default-vk-downward-down-volume:/etc/podinfo# cat metadata.labels
app="nginx"
env="test"
root@default-vk-downward-down-volume:/etc/podinfo#
```

4.Configure a security context

This topic describes how to configure a security context and define permissions and access control settings for a pod or container.

Background information

Security contexts are used to define permissions and access control settings for pods or containers. Security context settings include Discretionary Access Control, SELinux, and Linux Capabilities. For more information, see Configure a Security Context for a Pod or Container.

Kubernetes provides two methods to configure security contexts:

- Pod Security Context

  Configure a security context for a pod. The configured security context applies to all the containers and volumes in the pod.

  Elastic Container Instance allows you to modify the sysctl and runAsUse parameters when you configure a security context for a pod.

- Container Security Context

  Configure a security context for a container. The configured security context applies to the container.

  Elastic Container Instance allows you to modify the sysctl and runAsUse parameters when you configure a security context for a container.

> **Note**
>
> Kubernetes uses pod security policies to verify and restrict the security contexts of pods. If the security context of a pod to be created does not meet the constraints of the specified pod security policy, the pod cannot be created. For more information, see Pod Security Policies.
>
> If you use Container Service for Kubernetes (ACK) or Serverless Kubernetes (ASK), a pod security policy named ack.privileged is used by default. For more information, see Use pod security policies.

Configure a security context for a pod

In Linux, you can modify runtime kernel parameters by using the sysctl interface. You can view the kernel parameters of an elastic container instance by running the following command. For more information, see sysctl.sh.

```
sysctl -a
```

You can modify the sysctl and runAsUse parameters when you configure a security context for a pod.

The following sysctl parameters can be modified in Elastic Container Instance:

- kernel.shm* (except for kernel.shm_rmid_forced)
- kernel.msg*
- kernel.sem
- fs.mqueue.*

- net.* (except for net.ipv4.ip_local_port_range and net.ipv4.tcp_syncookies)

> **Warning**
>
> To avoid instability in the operating system, you must fully understand the impacts of sysctl parameter modifications before you proceed.

Sample code:

```
apiVersion: v1
kind: Pod
metadata:
  name: sysctl-example
spec:
  securityContext:
    sysctls:
    - name: net.core.somaxconn
      value: "1024"
    - name: kernel.msgmax
      value: "65536"
  containers:
  - name: busybox
    image: busybox
    command: [ "sh", "-c", "sleep 12000" ]
```

Configure a security context for a container

You can configure a security context for a specified container.

> **Note**
>
> You must specify some parameters such as runAsUse regardless of whether you configure a security context for a pod or for a container. The parameter settings in the security context of a container override those in the security context of the pod to which the container belongs to take effect on the pod.

The following table describes the parameters supported by Elastic Container Instance.

Parameter	Description
runAsUse	The ID of the user that runs the container. The parameter settings override the USER command in the Dockerfile.

Parameter	Description
capabilities	The permissions that are granted to processes in the container. For more information, see Linux capabilities. You can configure the following permissions: • AUDIT_WRITE • CHOWN • DAC_OVERRIDE • FSETID • FOWNER • KILL • MKNOD • NET_ADMIN • NET_BIND_SERVICE • NET_RAW • SETGID • SETUID • SETFCAP • SETPCAP • SYS_CHROOT • SYS_PTRACE • SYS_RAWIO Note You cannot grant SYS_RAWIO to processes by yourself. To use SYS_RAWIO, submit a ticket.

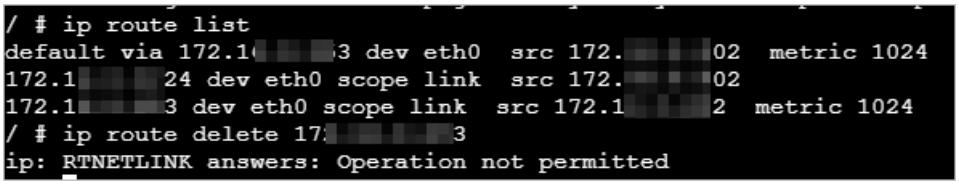
The following table describes some parameters that are not supported and the default values of the parameters.

Parameter	Description
privileged	Specifies whether to run the container in privileged mode. Default value: false.
AllowedProcMountTypes	The allowed proc mount types for the container. Default value: DefaultProcMount.

Parameter	Description
readOnlyRootFilesystem	Species whether the root file system that the container runs is read-only. Default value: true.

Sample code:

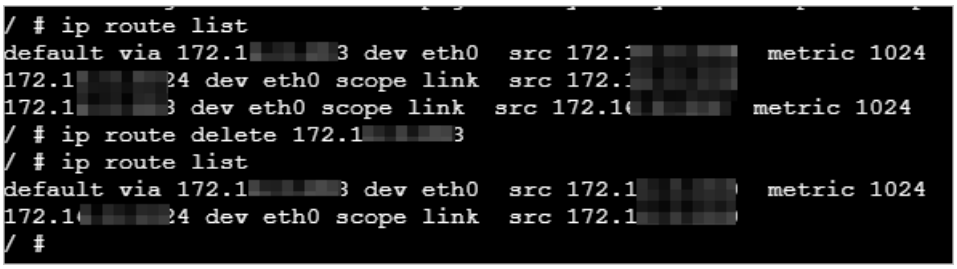
By default, containers do not have the NET_ADMIN permission. If network-related operations are performed in a container, an error message is returned.



You can configure the security context for the container and modify the capabilities parameter to add the NET_ADMIN permission. The following sample code provides an example on how to add the NET_ADMIN permission by modifying the capabilities parameter.

```
apiVersion: v1
kind: Pod
metadata:
  name: sysctl-example
spec:
  containers:
  - name: busybox
    image: busybox
    command: ["sh", "-c", "sleep 12000"]
    securityContext:
      capabilities:
        add: ["NET_ADMIN"]
```

After the permission is added, network-related operations can be performed in the container.



5.Enable the core dump feature

This topic describes how to enable the core dump feature. You can view and analyze a core dump file to identify the cause of the issue when a container is unexpectedly terminated.

Background information

In Linux, if a program unexpectedly terminates or exits, the operating system records the then state of the random access memory (RAM) that is allocated to the program and saves the state to a file. This process is called a core dump. You can view and analyze the core dump file to identify the cause of the issue.

The following figure shows the signals that indicate core dumps in Linux. The value of the signals in the Action column is Core.

Signal	Standard	Action	Comment
SIGABRT	P1990	Core	Abort signal from abort(3)
SIGALRM	P1990	Term	Timer signal from alarm(2)
SIGBUS	P2001	Core	Bus error (bad memory access)
SIGCHLD	P1990	Ign	Child stopped or terminated
SIGCLD	-	Ign	A synonym for SIGCHLD
SIGCONT	P1990	Cont	Continue if stopped
SIGEMT	-	Term	Emulator trap
SIGFPE	P1990	Core	Floating-point exception
SIGHUP	P1990	Term	Hangup detected on controlling terminal or death of controlling process
SIGILL	P1990	Core	Illegal Instruction
SIGINFO	-		A synonym for SIGPWR
SIGINT	P1990	Term	Interrupt from keyboard
SIGIO	-	Term	I/O now possible (4.2BSD)
SIGIOT	-	Core	IOT trap. A synonym for SIGABRT
SIGKILL	P1990	Term	Kill signal
SIGLOST	-	Term	File lock lost (unused)
SIGPIPE	P1990	Term	Broken pipe: write to pipe with no readers; see pipe(7)
SIGPOLL	P2001	Term	Pollable event (Sys V). Synonym for SIGIO
SIGPROF	P2001	Term	Profiling timer expired
SIGPWR	-	Term	Power failure (System V)
SIGQUIT	P1990	Core	Quit from keyboard
SIGSEGV	P1990	Core	Invalid memory reference
SIGSTKFLT	-	Term	Stack fault on coprocessor (unused)
SIGSTOP	P1990	Stop	Stop process
SIGTSTP	P1990	Stop	Stop typed at terminal
SIGSYS	P2001	Core	Bad system call (SVr4); see also seccomp(2)
SIGTERM	P1990	Term	Termination signal
SIGTRAP	P2001	Core	Trace/breakpoint trap
SIGTTIN	P1990	Stop	Terminal input for background process
SIGTTOU	P1990	Stop	Terminal output for background process
SIGUNUSED	-	Core	Synonymous with SIGSYS
SIGURG	P2001	Ign	Urgent condition on socket (4.2BSD)
SIGUSR1	P1990	Term	User-defined signal 1
SIGUSR2	P1990	Term	User-defined signal 2
SIGVTALRM	P2001	Term	Virtual alarm clock (4.2BSD)
SIGXCPU	P2001	Core	CPU time limit exceeded (4.2BSD); see setrlimit(2)
SIGXFSZ	P2001	Core	File size limit exceeded (4.2BSD); see setrlimit(2)

For more information, see Core dump file.

Overview

By default, the core dump feature is disabled in elastic container instances to prevent service unavailability due to excessive disk usage. You can use one of the following methods to enable the core dump feature based on your business requirements:

- Method 1: Execute a core dump O&M task

  After you enable the core dump feature, the system generates an O&M task. If a container unexpectedly terminates or exits, the core dump feature is triggered to generate a core file. The core file is automatically saved to an Object Storage Service (OSS) bucket.

- Method 2: Specify the storage path of core files

  Elastic Container Instance allows you to use an external storage device to store core files. After you specify the storage device, the core dump feature is automatically enabled for the elastic container instance. If a container unexpectedly terminates or exits, the core dump feature is triggered to generate a core file. The core file is saved to the specified external storage device.

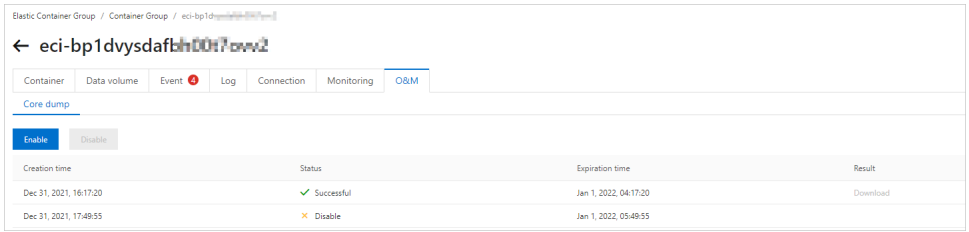
> **Note**
>
> - Method 1 is easy to use, but there are limits on the validity periods and regions that you specify. Method 1 is suitable for temporary program debugging and diagnosis.
>   - An O&M task that is generated by using Method 1 can be executed only once. After the O&M task is executed and a core file is obtained, the core dump feature is disabled. In addition, the O&M task has a valid period of 12 hours.
>   - Method 1 is based on OSS and Message Service (MNS). Method 1 cannot be used in the following regions because these regions support Elastic Container Instance but do not support OSS and MNS: China (Beijing), China (Ulanqab), China (Heyuan), China (Guangzhou), and China (Nanjing).
> - Method 2 requires an external storage device and ensures that a core file can be obtained even if the program is running in an unstable state. However, if issues occur in the program, repeatedly restarting the program may cause a large number of core files to be generated.

Method 1: Execute a core dump O&M task

Use the Elastic Container Instance console

1. Log on to the Elastic Container Instance console and create an elastic container instance.
2. Enable the core dump feature.
   i. Click the instance ID to open the instance details page.
   ii. Click the **O&M** tab, and then click **Enable**.

   After you click **Enable**, the system generates an O&M task. Before the core dump feature is triggered, the task is in the **Pending** state.



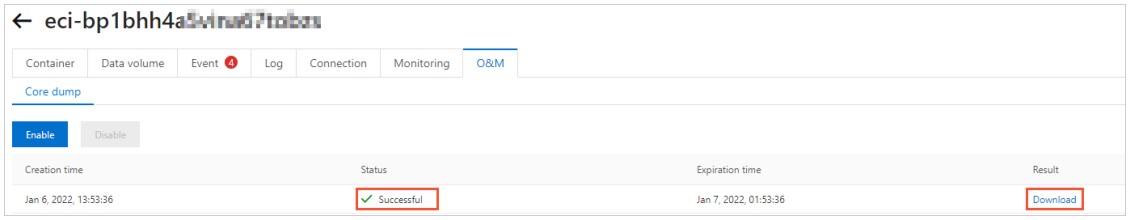
3. Trigger the core dump feature.

   After you run the following `sleep 100` command in the container, press the `Ctrl` key and the `\` key at the same time to trigger the core dump feature. A core file is automatically generated and saved to an OSS bucket.

```
# sleep 100
^\Quit (core dumped)
```

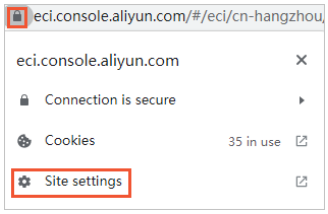
4. Download the core file.

   On the **O&M** tab of the instance details page, you can view the O&M task that is generated for the core dump. After the core dump feature is triggered and the system generates a core file, the status of the O&M task changes to **Successful**. Click **Download** in the Result column to download the core file to your on-premises computer.

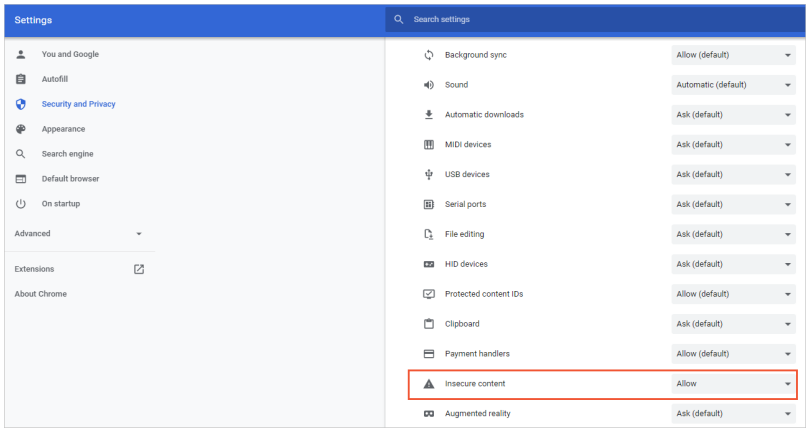


If the system does not respond, check the website permission settings of your browser. For example, if you use Google Chrome, you can use the following method to enable the download permissions:

i. In the Elastic Container Instance console, click the icon in the address bar of your browser, and then select **Site settings**.



ii. Change the settings of the configuration item **Insecure content** to **Allow**.



Call API operations

1. Create an elastic container instance.

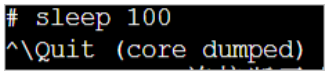
   When you call the CreateContainerGroup API operation to create an elastic container instance, do not specify the CorePattern parameter.

2. Enable the core dump feature.

   Call the CreateInstanceOpsTask API operation to create an O&M task. Set OpsType to `coredump` and OpsValue to `enable` to enable the core dump feature. For more information, see CreateInstanceOpsTask.

3. Trigger the core dump feature.

After you run the following `sleep 100` command in the container, press the `Ctrl` key and the `\` key at the same time to trigger the core dump feature. A core file is automatically generated and saved to an OSS bucket.



4. Download the core file.

   Call the DescribeInstanceOpsRecords API operation to query the result of the O&M task that is executed. You can view the ResultContent response parameter to obtain the URL of the path in which the core file is stored. Then, you can access this URL to download the core file.

Method 2: Specify the storage path of core files

Description

Elastic Container Instance allows you to specify the path in which core files are stored. After you specify the path, the core dump feature is automatically enabled. Core files are used to analyze issues offline. In most cases, core files are stored on external storage devices instead of local paths of containers to prevent the loss of core files if the containers exit. You can use one of the following methods to specify the storage path of core files:

> **Notice**
>
> The storage path cannot start with `|`. You cannot use core dump files to configure executable programs.

- Call API operations

  When you call the CreateContainerGroup API operation to create an elastic container instance, you can use the CorePattern parameter to specify the storage path of core files:

  ```
  CorePattern = "/xx/xx/core"
  ```

- Use Kubernetes environment

  When you create a pod in Kubernetes, you can add annotations to specify the storage path of core files.

```
apiVersion: apps/v1 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
  name: deployment-test
  labels:
    app: test
spec:
  replicas: 2
  selector:
    matchLabels:
      alibabacloud.com/eci: "true"
  template:
    metadata:
      labels:
        alibabacloud.com/eci: "true"
      annotations:
        k8s.aliyun.com/eci-core-pattern: "/xx/xx/core" # Specify the storage path of core
files.
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9 # replace it with your exactly <image_name:tags>
        ports:
        - containerPort: 80
```

Configuration example

In the following example, an API call method is used to configure a network-attached storage (NAS) file system as the external storage device of core files.

1. Create Elastic Container Instance A, mount the NAS file system, and specify the storage path of core files.

   When you call the CreateContainerGroup API operation to create Elastic Container Instance A, configure the following parameters to mount the `/dump/` directory of the NAS file system to the `/data/dump-a/` directory of the container and set the storage path of core files to `/data/dump/core`.

   ```
   'Volume.1.Name': 'volume1',
   'Volume.1.Type': 'NFSVolume',
   'Volume.1.NFSVolume.Path': '/dump/',
   'Volume.1.NFSVolume.Server': '143b24****-gfn3.cn-beijing.nas.aliyuncs.com',

   'Container.1.VolumeMount.1.Name': 'volume1',
   'Container.1.VolumeMount.1.MountPath': '/data/dump-a/',

   'CorePattern':'/data/dump-a/core',
   ```

2. Trigger the core dump feature in a directory of Elastic Container Instance A.

For example, after you run the `sleep 100` command in the container, you press the `Ctrl` key and the `\` key at the same time to trigger the core dump feature. The core file is saved to the `/data/dump-a/` path of the container.

```
# ls
bin  boot  data  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
# cd /data/dump-a/
# ls
# sleep 100
^\Quit (core dumped)
# ls
core.17
```

3. Release Elastic Container Instance A.
4. Mount the same NAS file system to Elastic Container Instance B.

   When you call the CreateContainerGroup API operation to create Elastic Container Instance B, specify the following parameters to mount the `/dump/` directory of the NAS file system to the `/data/dump-b/` directory of the container.

```
'Volume.1.Name': 'volume1',
'Volume.1.Type': 'NFSVolume',
'Volume.1.NFSVolume.Path': '/dump/',
'Volume.1.NFSVolume.Server': '143b24****-gfn3.cn-beijing.nas.aliyuncs.com',

'Container.1.VolumeMount.1.Name': 'dvolume1',
'Container.1.VolumeMount.1.MountPath': '/data/dump-b/',
```

5. View the core file in the container of Elastic Container Instance B.

   The following figure shows that the core file is displayed in the `/data/dump-b/` path of the container. After the core file is saved to the external storage device, the core file is not lost with the release of Elastic Container Instance A. You can view and analyze the contents of the core file.

```
# ls
bin  boot  data  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
# cd /data/dump-b/
# ls
core.17
```

6.Configure container termination messages

This topic describes how to set the terminationMessagePath and terminationMessagePolicy fields for a container. This allows you to customize the termination message of a container.

Kubernetes can configure the source of a container termination message by using terminationMessagePath. When a container is terminated, Kubernetes retrieves the termination message from the termination message file specified by the terminationMessagePath field of the container. Default value: /dev/termination-log.

You can customize the terminationMessagePath field of a container for Kubernetes to use the content of the specified custom file to fulfill the termination message of the container when the container running process completes or fails. The maximum size of a termination message is 4 KB.

In the following example, the terminationMessagePath field is configured as /tmp/termination-log. The container writes the termination message to the /tmp/termination-log file for Kubernetes to receive.

```
apiVersion: v1
kind: Pod
metadata:
  name: msg-path-demo
spec:
  containers:
  - name: msg-path-demo-container
    image: debian
    terminationMessagePath: "/tmp/termination-log"
```

In addition, you can set the terminationMessagePolicy field of the container to further customize the container termination message. Default value: File. This value indicates that termination messages can be retrieved only from the termination message file. You can set the terminationMessagePolicy field to FallbackToLogsOnError. This value indicates the last part of the container log output is used as the termination message if the termination message file is empty when the container exits due to exceptions.

```
apiVersion: v1
kind: Pod
metadata:
  name: msg-path-demo
spec:
  containers:
  - name: msg-path-demo-container
    image: debian
    terminationMessagePath: "/tmp/termination-log"
    terminationMessagePolicy: "FallbackToLogsOnError"
```

The total size of termination messages of all containers in a pod cannot exceed 12 KB. If the total size exceeds 12 KB, the state manager of Kubernetes sets a limit on the termination message sizes. For example, if a pod contains four InitContainers and eight application containers, the state manager limits the termination message of each container to 1 KB. This indicates that only the first 1 KB of the termination message of each container is intercepted.

7.Configure NTP for pods

This topic describes how to configure the Network Time Protocol (NTP) service for pods that run on a Virtual Kubelet node. The NTP service synchronizes the time between containers and the NTP server.

Prerequisites

Virtual Kubelet is upgraded to the latest version. For more information, see Update Virtual Kubelet.

Background information

Different types of Kubernetes clusters require different operations to update Virtual Kubelet.

- For Serverless Kubernetes (ASK) clusters, the system automatically updates Virtual Kubelet.
- For Container Service for Kubernetes (ACK) clusters, the update method of Virtual Kubelet varies based on the edition of the clusters. For managed ACK clusters, the system automatically updates Virtual Kubelet. For dedicated ACK clusters, you must manually update Virtual Kubelet.
- For self-managed Kubernetes clusters, you must manually update Virtual Kubelet.

Procedure

In the configuration file of the pod, add the following annotation to specify the IP address of the NTP server that you want to use: `k8s.aliyun.com/eci-ntp-server`.

1. Create a YAML file that is used to configure the NTP service.

   ```
   vim set-ntp-pod.yaml
   ```

   The content of the YAML file is:

   ```
   apiVersion: v1
   kind: Pod
   metadata:
     annotations:
       k8s.aliyun.com/eci-ntp-server: 10.10.5.1 # The IP address of the NTP server
     name: set-custom-ntp
   spec:
     nodeName: virtual-kubelet
     containers:
       - image: centos:latest
         command:
           - sleep
           - "3600"
         imagePullPolicy: IfNotPresent
         name: centos
   ```

2. Apply the configurations in the YAML file to the pod.

   ```
   kubectl apply -f set-ntp-pod.yaml
   ```

Check the result

Log on the elastic container instance to check whether the NTP service works as expected.

1. Query the information about the pod.

```
kubectl get pod/set-custom-ntp
```

Example:

```
NAME                 READY   STATUS    RESTARTS   AGE
set-custom-ntp   1/1         Running   0                7m20s
```

2. Go to the container.

```
kubectl exec set-custom-ntp -it -- bash
```

3. Query the source of the time of the container.

```
chronyc sources
```

If the IP address of the NTP server is returned, the NTP service works as expected. Example:

```
210 Number of sources = 1
MS Name/IP address         Stratum Poll Reach LastRx Last                            sa
mple
===============================================================================
^* 10.10.5.1                     2           6     377     35       +40us[ +135us] +/-
14ms
```

8.Configure a time zone for a pod

This topic describes how to configure different time zones for pods that run on Virtual Kubelet. When you use pods to deploy applications and want your pods to specify different time zones of different locations, you can refer to this topic.

Prerequisites

Virtual Kubelet is upgraded to the latest version.

Background information

The following list describes the methods to upgrade Virtual Kubelet to the latest version for different types of Kubernetes clusters:

- Serverless Kubernetes (ASK) clusters: The administrator upgrades Virtual Kubelet.
- Managed Kubernetes clusters: You must upgrade Virtual Kubelet by yourself.
- Dedicated Kubernetes clusters: You must upgrade Virtual Kubelet by yourself.
- Self-managed Kubernetes clusters: You must upgrade Virtual Kubelet by yourself.

Procedure

1. Create a ConfigMap and import the time zone that you want to specify.

   For other time zones, use the corresponding files in the /usr/share/zoneinfo/Asia/ directory. Example:

   ```
   kubectl create configmap tz --from-file=/usr/share/zoneinfo/Asia/Shanghai
   ```

2. Create a YAML file to configure the time zone.

   ```
   vim set-timezone.yaml
   ```

   Mount the ConfigMap to the /etc/localtime/Shanghai directory. Example:

```
apiVersion: v1
kind: Pod
metadata:
  name: timezone
spec:
  containers:
  - name: timezone
    image: registry-vpc.cn-beijing.aliyuncs.com/eci_open/busybox:1.30
    command: [ "sleep", "10000" ]
    volumeMounts:
      - name: tz
        mountPath: /etc/localtime
        subPath: Shanghai
  volumes:
    - name: tz
      configMap:
        name: tz
  nodeSelector:
    type: virtual-kubelet
  tolerations:
  - key: virtual-kubelet.io/provider
    operator: Exists
```

3. Apply the configurations in the YAML file to the pod.

```
kubectl apply -f set-timezone.yaml
```

Verify the result

Log on to the container and check whether the time zone is configured.

1. Obtain the pod information.

```
kubectl get pod/timezone
```

The following example shows the returned result:

```
NAME         READY   STATUS    RESTARTS   AGE
timezone     1/1     Running   0          7m20s
```

2. Go to the container.

```
kubectl exec timezone -it -- sh
```

3. Query the time zone of the container.

```
date -R
```

If the time returned corresponds to the time zone, the time zone is configured. The following example shows the returned result:

```
Fri, 01 May 2020 10:00:11 +0800
```