openEuler
20.03 LTS

Container User Guide

Date 2020-04-01

Contents

Terms of Use

Trademarks and Permissions

openEuler is a trademark or registered trademark of Huawei Technologies Co., Ltd. All other trademarks and registered trademarks mentioned in this document are the property of their respective holders.

Disclaimer

This document is used only as a guide. Unless otherwise specified by applicable laws or agreed by both parties in written form, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, including but not limited to non-infringement, timeliness, and specific purposes.

About This Document

Overview

The openEuler software package provides iSula, the basic platform for running containers.

iSula is a brand of Huawei's container technology solution. It originally means a kind of ant. This ant is also known as "bullet ant" due to the extremely painful sting, which has been compared to being shot by a bullet. In the eyes of Brazilian natives living in the Amazon jungle in Central and South America, iSula is one of the most powerful insects in the world. Huawei names the container technology solution brand based on its meaning.

The basic container platform iSula provides both Docker engine and lightweight container engine iSulad. You can select either of them as required.

In addition, the following container forms are provided on different application scenarios:

- Common containers applicable to most common scenarios
- Secure containers applicable to strong isolation and multi-tenant scenarios
- System containers applicable to scenarios where the systemd is used to manage services

This document describes how to install and use the container engines and how to deploy and use containers in different forms.

Intended Audience

This document is intended for openEuler users who need to install containers. You can better understand this document if you:

- Be familiar with basic Linux operations.
- Have a basic understanding of containers.

Symbol Conventions

The symbols that may be found in this document are defined as follows.

Symbol	Description
NOTICE	Indicates a potentially hazardous situation which, if not avoided, could result in equipment damage, data loss, performance deterioration, or unanticipated results. NOTICE is used to address practices not related to personal injury.

Symbol	Description
NOTE	Supplements the important information in the main text. NOTE is used to address information not related to personal injury, equipment damage, and environment deterioration.

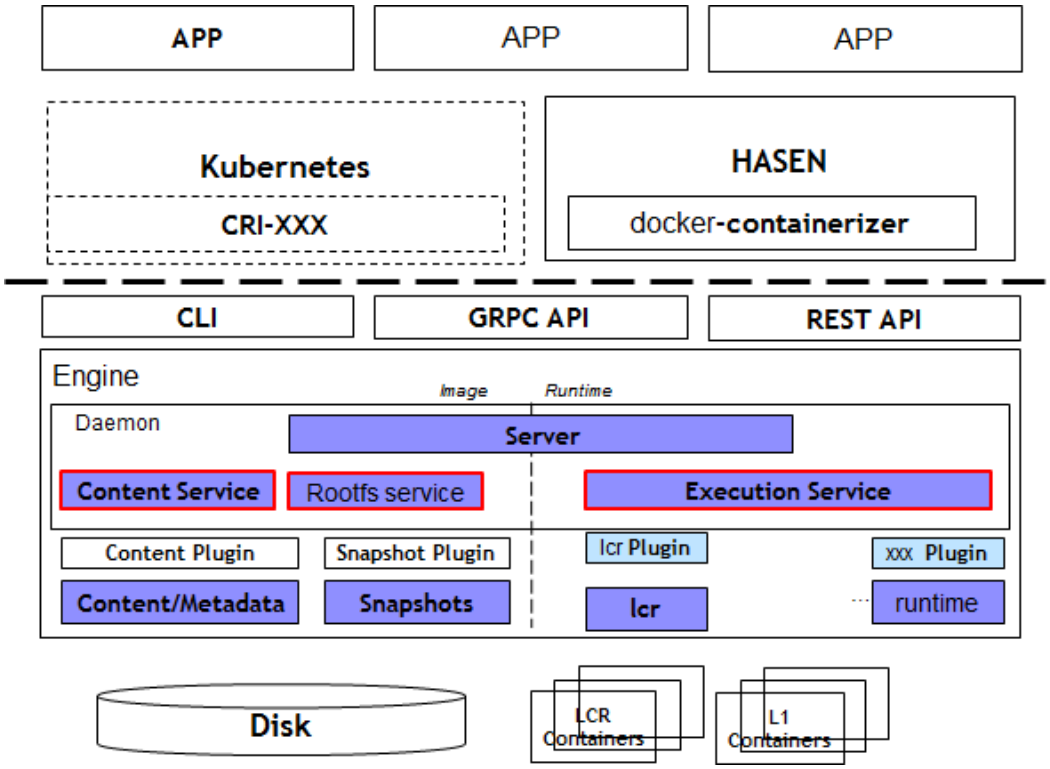
1 iSulad Container Engine

1.1 Overview

Compared with Docker, iSulad is a new container solution with a unified architecture design to meet different requirements in the CT and IT fields. Lightweight containers are implemented using C/C++. They are smart, fast, and not restricted by hardware and architecture. With less noise floor overhead, the containers can be widely used.

Figure 1-1 shows the unified container architecture.

Figure 1-1 Unified container architecture



1.2 Installation and Deployment

1.2.1 Installation Methods

iSulad can be installed by running the **yum** or **rpm** command. The **yum** command is recommended because dependencies can be installed automatically.

This section describes two installation methods.

- (Recommended) Run the following command to install iSulad:

```
$ sudo yum install -y iSulad
```

- If the **rpm** command is used to install iSulad, you need to download and manually install the RMP packages of iSulad and all its dependencies. To install the RPM package of a single iSulad (the same for installing dependency packages), run the following command:

```
$ sudo rpm -ihv iSulad-xx.xx.xx-YYYYmmdd.HHMMSS.gitxxxxxxxx.aarch64.rpm
```

1.2.2 Upgrade Methods

- For an upgrade between patch versions of a major version, for example, upgrading 2.x.x to 2.x.x, run the following command:

```
$ sudo yum update -y iSulad
```

- For an upgrade between major versions, for example, upgrading 1.x.x to 2.x.x, save the current configuration file **/etc/isulad/daemon.json**, uninstall the existing iSulad software package, install the iSulad software package to be upgraded, and restore the configuration file.

NOTE

- You can run the **sudo rpm -qa |grep iSulad** or **isula version** command to check the iSulad version.
- If you want to manually perform upgrade between patch versions of a major version, run the following command to download the RPM packages of iSulad and all its dependent libraries:

  ```
  $ sudo rpm -Uhv iSulad-xx.xx.xx-YYYYmmdd.HHMMSS.gitxxxxxxxx.aarch64.rpm
  ```

  If the upgrade fails, run the following command to forcibly perform the upgrade:

  ```
  $ sudo rpm -Uhv --force iSulad-xx.xx.xx-YYYYmmdd.HHMMSS.gitxxxxxxxx.aarch64.rpm
  ```

1.2.3 Deployment Configuration

1.2.3.1 Deployment Mode

The iSulad server daemon **isulad** can be configured with a configuration file or by running the **isulad --xxx** command. The priority in descending order is as follows: CLI > configuration file > default configuration in code.

NOTE

If systemd is used to manage the iSulad process, modify the **OPTIONS** field in the **/etc/sysconfig/iSulad** file, which functions the same as using the CLI.

- **CLI**

  During service startup, configure iSulad using the CLI. To view the configuration options, run the following command:

```
$ isulad --help
lightweight container runtime daemon

Usage:  isulad [global options]

GLOBAL OPTIONS:

     --authorization-plugin          Use authorization plugin
     --cgroup-parent                Set parent cgroup for all containers
     --cni-bin-dir                  The full path of the directory in which to search
for CNI plugin binaries. Default: /opt/cni/bin
     --cni-conf-dir                 The full path of the directory in which to
search for CNI config files. Default: /etc/cni/net.d
     --default-ulimit               Default ulimits for containers (default [])
  -e, --engine                      Select backend engine
  -g, --graph                       Root directory of the iSulad runtime
  -G, --group                       Group for the unix socket(default is isulad)
     --help                         Show help
     --hook-spec                    Default hook spec file applied to all containers
  -H, --host                        The socket name used to create gRPC server
     --image-layer-check             Check layer intergrity when needed
     --image-opt-timeout             Max timeout(default 5m) for image operation
     --insecure-registry             Disable TLS verification for the given
registry
     --insecure-skip-verify-enforce   Force to skip the insecure verify(default
false)
     --log-driver                   Set daemon log driver, such as: file
  -l, --log-level                    Set log level, the levels can be: FATAL ALERT
CRIT ERROR WARN NOTICE INFO DEBUG TRACE
     --log-opt                      Set daemon log driver options, such as:
log-path=/tmp/logs/ to set directory where to store daemon logs
```

```
     --native.umask                 Default file mode creation mask (umask) for
containers
     --network-plugin               Set network plugin, default is null, suppport
null and cni
  -p, --pidfile                     Save pid into this file
     --pod-sandbox-image             The image whose network/ipc namespaces
containers in each pod will use. (default
"rnd-dockerhub.huawei.com/library/pause-${machine}:3.0")
     --registry-mirrors              Registry to be prepended when pulling
unqualified images, can be specified multiple times
     --start-timeout                timeout duration for waiting on a container to
start before it is killed
  -S, --state                       Root directory for execution state files
     --storage-driver                Storage driver to use(default overlay2)
  -s, --storage-opt                  Storage driver options
     --tls                          Use TLS; implied by --tlsverify
     --tlscacert                     Trust certs signed only by this CA (default
"/root/.iSulad/ca.pem")
     --tlscert                      Path to TLS certificate file (default
"/root/.iSulad/cert.pem")
     --tlskey                       Path to TLS key file (default
"/root/.iSulad/key.pem")
     --tlsverify                    Use TLS and verify the remote
     --use-decrypted-key             Use decrypted private key by default(default
true)
  -V, --version                     Print the version
   --websocket-server-listening-port   CRI websocket streaming service listening
port (default 10350)
```

Example: Start iSulad and change the log level to DEBUG.

```
$ isulad -l DEBUG
```

- **Configuration file**

  The iSulad configuration file is **/etc/isulad/daemon.json**. The parameters in the file are described as follows:

Parameter	Example	Description	Remarks
-e, --engine	"engine": "lcr"	iSulad runtime, which is **lcr** by default.	None
-G, --group	"group": "isulad"	Socket group.	None
--hook-spec	"hook-spec": "/etc/default/isulad/hooks/default.json"	Default hook configuration file for all containers.	None
-H, --host	"hosts": "unix:///var/run/isulad.sock"	Communication mode.	In addition to the local socket, the **tcp://ip:port** mode is supported. The port number ranges from 0 to 65535, excluding occupied ports.

Parameter	Example	Description	Remarks
--log-driver	"log-driver": "file"	Log driver configuration.	None
-l, --log-level	"log-level": "ERROR"	Log output level.	None

--log-opt	"log-opts": {
	"log-file-mode": "0600",
	"log-path": "/var/lib/isulad",
	"max-file": "1",
	"max-size": "30KB"
	}
		Log-related configuration.	You can specify **max-file**, **max-size**, and **log-path**. **max-file** indicates the number of log files. **max-size** indicates the threshold for triggering log anti-explosion. If **max-file** is **1**, **max-size** is invalid. **log-path** specifies the path for storing log files. The **log-file-mode** command is used to set the permissions to read and write log files. The value must be in octal format, for example, 0666.

--start-timeout	"start-timeout": "2m"	Time required for starting a container.	None
--runtime	"default-runtime": "lcr"	Container runtime, which is **lcr** by default.	If neither the CLI nor the configuration file specifies the runtime, **lcr** is used by default. The priorities of the three specifying methods are as follows: CLI > configuration file > default value **lcr**. Currently, **lcr** and **kata-runtime** are supported.

None	`"runtimes": {`
	`"kata-runtime": {`
	`"path": "/usr/bin/kata-runtime",`
	`"runtime-args": [`
	`"--kata-config", "/usr/share/defaults/kata-containers/configuration.toml"`
	`]`
	`}`
	`}`
		When starting a container, set this parameter to specify multiple runtimes. Runtimes in this set are valid for container startup.	Runtime whitelist of a container. The customized runtimes in this set are valid. **kata-runtime** is used as the example.

-p,	"pidfile":	File for storing	This parameter is required

Parameter	Example	Description	Remarks
--pidfile	"/var/run/isulad.pid"	PIDs.	only when more than two container engines need to be started.
-g, --graph	"graph": "/var/lib/isulad"	Root directory for iSulad runtimes.	
-S, --state	"state": "/var/run/isulad"	Root directory of the execution file.	
--storage-driver	"storage-driver": "overlay2"	Image storage driver, which is **overlay2** by default.	Only **overlay2** is supported.

-s, --storage-opt	"storage-opts": ["overlay2.override_kernel_check=true"]	Image storage driver configuration options.	The options are as follows:
			`overlay2.override kernel check=true #Ignore the kernel version check.`
			`overlay2.size=${size} #Set the rootfs quota to ${size}.`
			`overlay2.basesize=${size} #It is equivalent to overlay2.size.`

--image-opt-timeout	"image-opt-timeout": "5m"	Image operation timeout interval, which is **5m** by default.	The value **-1** indicates that the timeout interval is not limited.
--registry-mirrors	"registry-mirrors": ["docker.io"]	Registry address.	None
--insecure-registry	"insecure-registries": [ ]	Registry without TLS verification.	None

--native.umask	"native.umask": "secure"	Container umask policy. The default value is **secure**. The value **normal** indicates insecure configuration.	Set the container **umask** value.
			The value can be null (**0027** by default), **normal**, or **secure**.
			`normal #The umask value of the started container is 0022.`
			`secure #The umask value of the started container is 0027 (default value).`

--pod-sandbox-image	"pod-sandbox-image": "rnd-dockerhub.huawei.com/library/pause-aarch64:3.0	By default, the pod uses the image. The default value is	None

Parameter	Example	Description	Remarks
	"	**rnd-dockerhub.huawei.com/library/pause-${machine}:3.0**.	
--network-plugin	"network-plugin": ""	Specifies a network plug-in. The value is a null character by default, indicating that no network configuration is available and the created sandbox has only the loop NIC.	The CNI and null characters are supported. Other invalid values will cause iSulad startup failure.
--cni-bin-dir	"cni-bin-dir": ""	Specifies the storage location of the binary file on which the CNI plug-in depends.	The default value is **/opt/cni/bin**.
--cni-conf-dir	"cni-conf-dir": ""	Specifies the storage location of the CNI network configuration file.	The default value is **/etc/cni/net.d**.
--image-layer-check=false	"image-layer-check": false	Image layer integrity check. To enable the function, set it to **true**; otherwise, set it to **false**. It is disabled by default.	When iSulad is started, the image layer integrity is checked. If the image layer is damaged, the related images are unavailable. iSulad cannot verify empty files, directories, and link files. Therefore, if the preceding files are lost due to a power failure, the integrity check of iSulad image data may fail to be identified. When the iSulad version changes, check whether the parameter is supported. If not, delete it from the configuration file.
--insecure-skip-verify	"insecure-skip-verify-enforc	Indicates whether to	The default value is **false** (not skipped). Note:

Parameter	Example	Description	Remarks
-enforce=false	e": false	forcibly skip the verification of the certificate host name/domain name. The value is of the Boolean type, and the default value is **false**. If this parameter is set to **true**, the verification of the certificate host name/domain name is skipped.	Restricted by the YAJL JSON parsing library, if a non-Boolean value that meets the JSON format requirements is configured in the **/etc/isulad/daemon.json** configuration file, the default value used by iSulad is **false**.
--use-decrypted-key=true	"use-decrypted-key": true	Specifies whether to use an unencrypted private key. The value is of the Boolean type. If this parameter is set to **true**, an unencrypted private key is used. If this parameter is set to **false**, the encrypted private key is used, that is, two-way authentication is required.	The default value is **true**, indicating that an unencrypted private key is used. Note: Restricted by the YAJL JSON parsing library, if a non-Boolean value that meets the JSON format requirements is configured in the **/etc/isulad/daemon.json** configuration file, the default value used by iSulad is **true**.
--tls	"tls":false	Specifies whether to use TLS. The value is of the Boolean type.	This parameter is used only in **-H tcp://IP:PORT** mode. The default value is **false**.
--tlsverify	"tlsverify":false	Specifies whether to use TLS and verify remote access. The value is of the Boolean type.	This parameter is used only in **-H tcp://IP:PORT** mode.
--tlscacert --tlscert	"tls-config": { "CAFile":	TLS certificate-relate	This parameter is used only in **-H tcp://IP:PORT**

Parameter	Example	Description	Remarks
--tlskey	"/root/.iSulad/ca.pem", "CertFile": "/root/.iSulad/server-cert.pem", "KeyFile":"/root/.iSulad/server-key.pem" }	d configuration.	mode.
--authorization-plugin	"authorization-plugin": "authz-broker"	User permission authentication plugin.	Only **authz-broker** is supported.
--cgroup-parent	"cgroup-parent": "lxc/mycgroup"	Default cgroup parent path of a container, which is of the string type.	Specifies the cgroup parent path of a container. If **--cgroup-parent** is specified on the client, the client parameter prevails. Note: If container A is started before container B, the cgroup parent path of container B is specified as the cgroup path of container A. When deleting a container, you need to delete container B and then container A in sequence. Otherwise, residual cgroup resources exist.

--default-ulimits	"default-ulimits": {
	"nofile": {
	"Name": "nofile",
	"Hard": 6400,
	"Soft": 3200
	}
	}
		Specifies the ulimit restriction type, soft value, and hard value.	Specifies the restricted resource type, for example, nofile. The two field names must be the same, that is, nofile. Otherwise, an error is reported. The value of **Hard** must be greater than or equal to that of **Soft**. If the **Hard** or **Soft** field is not set, the default value **0** is used.

--websocket-server-listening-port	"websocket-server-listening-port": 10350	Specifies the listening port of the CRI WebSocket streaming service. The default port number is **10350**.	Specifies the listening port of the CRI websocket streaming service. If the client specifies **--websocket-server-listening-port**, the specified value is used. The port number ranges from 1024

Parameter	Example	Description	Remarks
			to 49151.

Example:

```
$ cat /etc/isulad/daemon.json
{
    "group": "isulad",
    "default-runtime": "lcr",
    "graph": "/var/lib/isulad",
    "state": "/var/run/isulad",
    "engine": "lcr",
    "log-level": "ERROR",
    "pidfile": "/var/run/isulad.pid",
    "log-opts": {
        "log-file-mode": "0600",
        "log-path": "/var/lib/isulad",
        "max-file": "1",
        "max-size": "30KB"
    },
    "log-driver": "stdout",
    "hook-spec": "/etc/default/isulad/hooks/default.json",
    "start-timeout": "2m",
    "storage-driver": "overlay2",
    "storage-opts": [
        "overlay2.override_kernel_check=true"
    ],
    "registry-mirrors": [
        "docker.io"
    ],
    "insecure-registries": [
        "rnd-dockerhub.huawei.com"
    ],
    "pod-sandbox-image": "",
    "image-opt-timeout": "5m",
    "native.umask": "secure",
    "network-plugin": "",
    "cni-bin-dir": "",
    "cni-conf-dir": "",
    "image-layer-check": false,
    "use-decrypted-key": true,
    "insecure-skip-verify-enforce": false
}
```

NOTICE

The default configuration file **/etc/isulad/daemon.json** is for reference only. Configure it based on site requirements.

1.2.3.2 Storage Description

File	Directory	Description
*	/etc/default/isulad/	Stores the OCI configuration file and hook template file of iSulad. The file configuration permission is set to **0640**, and the sysmonitor check permission is set to **0550**.
*	/etc/isulad/	Default configuration files of iSulad and seccomp.
isulad.sock	/var/run/	Pipe communication file, which is used for the communication between the client and iSulad.
isulad.pid	/var/run/	File for storing the iSulad PIDs. It is also a file lock to prevent multiple iSulad instances from being started.
*	/run/lxc/	Lock file, which is created during iSulad running.
*	/var/run/isulad/	Real-time communication cache file, which is created during iSulad running.
*	/var/run/isula/	Real-time communication cache file, which is created during iSulad running.
*	/var/lib/lcr/	Temporary directory of the LCR component.
*	/var/lib/isulad/	Root directory where iSulad runs, which stores the created container configuration, default log path, database file, and mount point. **/var/lib/isulad/mnt/**: mount point of the container rootfs. **/var/lib/isulad/engines/lcr/**: directory for storing LCR container configurations. Each container has a directory named after the container.

1.2.3.3 Constraints

- In high concurrency scenarios (200 containers are concurrently started), the memory management mechanism of Glibc may cause memory holes and large virtual memory (for example, 10 GB). This problem is caused by the restriction of the Glibc memory management mechanism in the high concurrency scenario, but not by memory leakage. Therefore, the memory consumption does not increase infinitely. You can set **MALLOC_ARENA_MAX** to reducevirtual memory error and increase the rate of reducing physical memory. However, this environment variable will cause the iSulad concurrency performance to deteriorate. Set this environment variable based on the site requirements.

  ```
  To balance performance and memory usage, set MALLOC_ARENA_MAX to 4. (The iSulad performance on the ARM64 server is affected by less than 10%.)

  Configuration method:
  1. To manually start iSulad, run the export MALLOC_ARENA_MAX=4 command and then start iSulad.
  2. If systemd manages iSulad, you can modify the /etc/sysconfig/iSulad file by adding MALLOC_ARENA_MAX=4.
  ```

- Precautions for specifying the daemon running directories

  Take **--root** as an example. When **/new/path/** is used as the daemon new root directory, if a file exists in **/new/path/** and the directory or file name conflicts with that required by iSulad (for example, **engines** and **mnt**), iSulad may update the original directory or file attributes including the owner and permission.

  Therefore, please note the impact of re-specifying various running directories and files on their attributes. You are advised to specify a new directory or file for iSulad to avoid file attribute changes and security issues caused by conflicts.

- Log file management:

NOTICE

Log function interconnection: logs are managed by systemd as iSulad is and then transmitted to rsyslogd. By default, rsyslog restricts the log writing speed. You can add the configuration item **$imjournalRatelimitInterval 0** to the **/etc/rsyslog.conf** file and restart the rsyslogd service.

- Restrictions on command line parameter parsing

  When the iSulad command line interface is used, the parameter parsing mode is slightly different from that of Docker. For flags with parameters in the command line, regardless of whether a long or short flag is used, only the first space after the flag or the character string after the equal sign (=) directly connected to the flag is used as the flag parameter. The details are as follows:

  a. When a short flag is used, each character in the character string connected to the hyphen (-) is considered as a short flag. If there is an equal sign (=), the character string following the equal sign (=) is considered as the parameter of the short flag before the equal sign (=).

     isula run -du=root busybox is equivalent to **isula run -du root busybox**, **isula run -d -u=root busybox**, or **isula run -d -u root busybox**. When **isula run -du:root** is used, as **-:** is not a valid short flag, an error is reported. The preceding command is equivalent to **isula run -ud root busybox**. However, this method is not recommended because it may cause semantic problems.

b. When a long flag is used, the character string connected to -- is regarded as a long flag. If the character string contains an equal sign (=), the character string before the equal sign (=) is a long flag, and the character string after the equal sign (=) is a parameter.

```
isula run --user=root busybox
```

or

```
isula run --user root busybox
```

- After an iSulad container is started, you cannot run the **isula run -i/-t/-ti** and **isula attach/exec** commands as a non-root user.
- When iSulad connects to an OCI container, only kata-runtime can be used to start the OCI container.

1.2.3.4 Daemon Multi-Port Binding

Description

The daemon can bind multiple UNIX sockets or TCP ports and listen on these ports. The client can interact with the daemon through these ports.

Port

Users can configure one or more ports in the hosts field in the **/etc/isulad/daemon.json** file, or choose not to specify hosts.

```
{
   "hosts": [
      "unix:///var/run/isulad.sock",
      "tcp://localhost:5678",
      "tcp://127.0.0.1:6789"
   ]
}
```

Users can also run the **-H** or **--host** command in the **/etc/sysconfig/iSulad** file to configure a port, or choose not to specify hosts.

```
OPTIONS='-H unix:///var/run/isulad.sock --host tcp://127.0.0.1:6789'
```

If hosts are not specified in the **daemon.json** file and iSulad, the daemon listens on **unix:///var/run/isulad.sock** by default after startup.

Restrictions

- Users cannot specify hosts in the **/etc/isulad/daemon.json** and **/etc/sysconfig/iSuald** files at the same time. Otherwise, an error will occur and iSulad cannot be started.

```
unable to configure the isulad with file /etc/isulad/daemon.json: the following directives are specified both as a flag and in the configuration file: hosts: (from flag: [unix:///var/run/isulad.sock tcp://127.0.0.1:6789], from file: [unix:///var/run/isulad.sock tcp://localhost:5678 tcp://127.0.0.1:6789])
```

- If the specified host is a UNIX socket, the socket must start with **unix://** followed by a valid absolute path.
- If the specified host is a TCP port, the TCP port number must start with **tcp://** followed by a valid IP address and port number. The IP address can be that of the local host.

- A maximum of 10 valid ports can be specified. If more than 10 ports are specified, an error will occur and iSulad cannot be started.

1.2.3.5 Configuring TLS Authentication and Enabling Remote Access

Description

iSulad is designed in C/S mode. By default, the iSulad daemon process listens only on the local/var/run/isulad.sock. Therefore, you can run commands to operate containers only on the local client iSula. To enable iSula's remote access to the container, the iSulad daemon process needs to listen on the remote access port using TCP/IP. However, listening is performed only by simply configuring tcp ip:port. In this case, all IP addresses can communicate with iSulad by calling **isula -H tcp://***remote server IP address***:port**, which may cause security problems. Therefore, it is recommended that a more secure version, namely Transport Layer Security (TLS), be used for remote access.

Generating TLS Certificate

- Example of generating a plaintext private key and certificate

```
#!/bin/bash
set -e
echo -n "Enter pass phrase:"
read password
echo -n "Enter public network ip:"
read publicip
echo -n "Enter host:"
read HOST

echo " => Using hostname: $publicip, You MUST connect to iSulad using this host!"

mkdir -p $HOME/.iSulad
cd $HOME/.iSulad
rm -rf $HOME/.iSulad/*

echo " => Generating CA key"
openssl genrsa -passout pass:$password -aes256 -out ca-key.pem 4096
echo " => Generating CA certificate"
openssl req -passin pass:$password -new -x509 -days 365 -key ca-key.pem -sha256 -out
ca.pem -subj
"/C=CN/ST=zhejiang/L=hangzhou/O=Huawei/OU=iSulad/CN=iSulad@huawei.com"
echo " => Generating server key"
openssl genrsa -passout pass:$password -out server-key.pem 4096
echo " => Generating server CSR"
openssl req -passin pass:$password -subj /CN=$HOST -sha256 -new -key server-key.pem
-out server.csr
echo subjectAltName = DNS:$HOST,IP:$publicip,IP:127.0.0.1 >> extfile.cnf
echo extendedKeyUsage = serverAuth >> extfile.cnf
echo " => Signing server CSR with CA"
openssl x509 -req -passin pass:$password -days 365 -sha256 -in server.csr -CA ca.pem
-CAkey ca-key.pem -CAcreateserial -out server-cert.pem -extfile extfile.cnf
echo " => Generating client key"
openssl genrsa -passout pass:$password -out key.pem 4096
echo " => Generating client CSR"
openssl req -passin pass:$password -subj '/CN=client' -new -key key.pem -out
client.csr
```

```
echo " => Creating extended key usage"
echo extendedKeyUsage = clientAuth > extfile-client.cnf
echo " => Signing client CSR with CA"
openssl x509 -req -passin pass:$password -days 365 -sha256 -in client.csr -CA ca.pem
-CAkey ca-key.pem -CAcreateserial -out cert.pem -extfile extfile-client.cnf
rm -v client.csr server.csr extfile.cnf extfile-client.cnf
chmod -v 0400 ca-key.pem key.pem server-key.pem
chmod -v 0444 ca.pem server-cert.pem cert.pem
```

- Example of generating an encrypted private key and certificate request file

```
#!/bin/bash

echo -n "Enter public network ip:"
read publicip
echo -n "Enter pass phrase:"
read password

# remove certificates from previous execution.
rm -f *.pem *.srl *.csr *.cnf



# generate CA private and public keys
echo 01 > ca.srl
openssl genrsa -aes256 -out ca-key.pem -passout pass:$password 2048
openssl req -subj
'/C=CN/ST=zhejiang/L=hangzhou/O=Huawei/OU=iSulad/CN=iSulad@huawei.com' -new
-x509 -days $DAYS -passin pass:$password -key ca-key.pem -out ca.pem

# create a server key and certificate signing request (CSR)
openssl genrsa -aes256 -out server-key.pem -passout pass:$PASS 2048
openssl req -new -key server-key.pem -out server.csr -passin pass:$password -subj
'/CN=iSulad'

echo subjectAltName = DNS:iSulad,IP:${publicip},IP:127.0.0.1 > extfile.cnf
echo extendedKeyUsage = serverAuth >> extfile.cnf
# sign the server key with our CA
openssl x509 -req -days $DAYS -passin pass:$password -in server.csr -CA ca.pem -CAkey
ca-key.pem -out server-cert.pem -extfile extfile.cnf

# create a client key and certificate signing request (CSR)
openssl genrsa -aes256 -out key.pem -passout pass:$password 2048
openssl req -subj '/CN=client' -new -key key.pem -out client.csr -passin
pass:$password

# create an extensions config file and sign
echo extendedKeyUsage = clientAuth > extfile.cnf
openssl x509 -req -days 365 -passin pass:$password -in client.csr -CA ca.pem -CAkey
ca-key.pem -out cert.pem -extfile extfile.cnf

# remove the passphrase from the client and server key
openssl rsa -in server-key.pem -out server-key.pem -passin pass:$password
openssl rsa -in key.pem -out key.pem -passin pass:$password

# remove generated files that are no longer required
rm -f ca-key.pem ca.srl client.csr extfile.cnf server.csr
```

APIs

```
{
   "tls": true,
   "tls-verify": true,
   "tls-config": {
        "CAFile": "/root/.iSulad/ca.pem",
        "CertFile": "/root/.iSulad/server-cert.pem",
        "KeyFile":"/root/.iSulad/server-key.pem"
   }
}
```

Restrictions

The server supports the following modes:

- Mode 1 (client verified): tlsverify, tlscacert, tlscert, tlskey
- Mode 2 (client not verified): tls, tlscert, tlskey

The client supports the following modes:

- Mode 1 (verify the identity based on the client certificate, and verify the server based on the specified CA): tlsverify, tlscacert, tlscert, tlskey
- Mode 2 (server verified): tlsverify, tlscacert

Mode 1 is used for the server, and mode 2 for the client if the two-way authentication mode is used for communication.

Mode 2 is used for the server and the client if the unidirectional authentication mode is used for communication.

NOTICE

- If RPM is used for installation, the server configuration can be modified in the **/etc/isulad/daemon.json** and **/etc/sysconfig/iSulad** files.
- Two-way authentification is recommended as it is more secure than non-authentication or unidirectional authentication.
- GRPC open-source component logs are not taken over by iSulad. To view gRPC logs, set the environment variables **gRPC_VERBOSITY** and **gRPC_TRACE** as required.

Example

On the server:

```
 isulad -H=tcp://0.0.0.0:2376 --tlsverify --tlscacert ~/.iSulad/ca.pem --tlscert
~/.iSulad/server-cert.pem --tlskey ~/.iSulad/server-key.pem
```

On the client:

```
 isula version -H=tcp://$HOSTIP:2376 --tlsverify --tlscacert ~/.iSulad/ca.pem
--tlscert ~/.iSulad/cert.pem --tlskey ~/.iSulad/key.pem
```

1.2.3.6 devicemapper Storage Driver Configuration

To use the devicemapper storage driver, you need to configure a thinpool device which requires an independent block device with sufficient free space. Take the independent block device **/dev/xvdf** as an example. The configuration method is as follows:

1. Configuring a thinpool

1. Stop the iSulad service.

```
# systemctl stop isulad
```

2. Create a logical volume manager (LVM) volume based on the block device.

```
# pvcreate /dev/xvdf
```

3. Create a volume group based on the created physical volume.

```
# vgcreate isula /dev/xvdf
Volume group "isula" successfully created:
```

4. Create two logical volumes named **thinpool** and **thinpoolmeta**.

```
# lvcreate --wipesignatures y -n thinpool isula -l 95%VG
Logical volume "thinpool" created.
# lvcreate --wipesignatures y -n thinpoolmeta isula -l 1%VG
Logical volume "thinpoolmeta" created.
```

5. Convert the two logical volumes into a thinpool and the metadata used by the thinpool.

```
# lvconvert -y --zero n -c 512K --thinpool isula/thinpool --poolmetadata
isula/thinpoolmeta

WARNING: Converting logical volume isula/thinpool and isula/thinpoolmeta to
thin pool's data and metadata volumes with metadata wiping.
THIS WILL DESTROY CONTENT OF LOGICAL VOLUME (filesystem etc.)
Converted isula/thinpool to thin pool.
```

2. Modifying the iSulad configuration files

6. If iSulad has been used in the environment, back up the running data first.

```
# mkdir /var/lib/isulad.bk
# mv /var/lib/isulad/* /var/lib/isulad.bk
```

7. Modify configuration files.

   Two configuration methods are provided. Select one based on site requirements.

   - Edit the **/etc/isulad/daemon.json** file, set **storage-driver** to **devicemapper**, and set parameters related to the **storage-opts** field. For details about related parameters, see Parameter Description. The following lists the configuration reference:

```
{
    "storage-driver": "devicemapper"
    "storage-opts": [
        "dm.thinpooldev=/dev/mapper/isula-thinpool",
        "dm.fs=ext4",
          "dm.min free space=10%"
    ]
}
```

   - You can also edit **/etc/sysconfig/iSulad** to explicitly specify related iSulad startup parameters. For details about related parameters, see Parameter Description. The following lists the configuration reference:

```
OPTIONS="--storage-driver=devicemapper --storage-opt
dm.thinpooldev=/dev/mapper/isula-thinpool --storage-opt dm.fs=ext4
--storage-opt dm.min_free_space=10%"
```

8. Start iSulad for the settings to take effect.

```
# systemctl start isulad
```

Parameter Description

For details about parameters supported by storage-opts, see Table 1-1.

Table 1-1 Parameter description

Parameter	Mandatory or Not	Description
dm.fs	Yes	Specifies the type of the file system used by a container. This parameter must be set to **ext4**, that is, **dm.fs=ext4**.
dm.basesize	No	Specifies the maximum storage space of a single container. The unit can be **k**, **m**, **g**, **t**, or **p**. An uppercase letter can also be used, for example, **dm.basesize=50G**. This parameter is valid only during the first initialization.
dm.mkfsarg	No	Specifies the additional **mkfs** parameters when a basic device is created. For example: **dm.mkfsarg=-O ^has_journal**
dm.mountopt	No	Specifies additional **mount** parameters when a container is mounted. For example: **dm.mountopt=nodiscard**
dm.thinpooldev	No	Specifies the thinpool device used for container or image storage.
dm.min_free_space	No	Specifies minimum percentage of reserved space. For example, **dm.min_free_space=10%** indicates that storage-related operations such as container creation will fail when the remaining storage space falls below 10%.

Precautions

- When configuring devicemapper, if the system does not have sufficient space for automatic capacity expansion of thinpool, disable the automatic capacity expansion function.

  To disable automatic capacity expansion, set both **thin_pool_autoextend_threshold** and **thin_pool_autoextend_percent** in the **/etc/lvm/profile/isula-thinpool.profile** file to **100**.

```
activation {
  thin_pool_autoextend_threshold=100
```

```
  thin pool autoextend percent=100
}
```

- When devicemapper is used, use Ext4 as the container file system. You need to add **--storage-opt dm.fs=ext4** to the iSulad configuration parameters.
- If graphdriver is devicemapper and the metadata files are damaged and cannot be restored, you need to manually restore the metadata files. Do not directly operate or tamper with metadata of the devicemapper storage driver in Docker daemon.
- When the devicemapper LVM is used, if the devicemapper thinpool is damaged due to abnormal power-off, you cannot ensure the data integrity or whether the damaged thinpool can be restored. Therefore, you need to rebuild the thinpool.

Precautions for Switching the devicemapper Storage Pool When the User Namespace Feature Is Enabled on iSula

- Generally, the path of the deviceset-metadata file is **/var/lib/isulad/devicemapper/metadata/deviceset-metadata** during container startup.
- If user namespaces are used, the path of the deviceset-metadata file is **/var/lib/isulad/***userNSUID.GID***/devicemapper/metadata/deviceset-metadata**.
- When you use the devicemapper storage driver and the container is switched between the user namespace scenario and common scenario, the **BaseDeviceUUID** content in the corresponding deviceset-metadata file needs to be cleared. In the thinpool capacity expansion or rebuild scenario, you also need to clear the **BaseDeviceUUID** content in the deviceset-metadata file. Otherwise, the iSulad service fails to be restarted.

1.2.4 Uninstallation

To uninstall iSulad, perform the following operations:

1. Uninstall iSulad and its dependent software packages.
   - If the **yum** command is used to install iSulad, run the following command to uninstall iSulad:

     ```
     $ sudo yum remove iSulad
     ```

   - If the **rpm** command is used to install iSulad, uninstall iSulad and its dependent software packages. Run the following command to uninstall an RPM package.

     ```
     sudo rpm -e iSulad-xx.xx.xx-YYYYmmdd.HHMMSS.gitxxxxxxxx.aarch64.rpm
     ```

2. Images, containers, volumes, and related configuration files are not automatically deleted. The reference command is as follows:

   ```
   $ sudo rm -rf /var/lib/iSulad
   ```

1.3 Application Scenarios

1.3.1 Container Management

1.3.1.1 Creating a Container

Description

To create a container, run the **isula create** command. The container engine will use the specified container image to create a read/write layer, or use the specified local rootfs as the

running environment of the container. After the creation is complete, the container ID is output as standard output. You can run the **isula start** command to start the container. The new container is in the **inited** state.

Usage

```
isula create [OPTIONS] IMAGE [COMMAND] [ARG...]
```

Parameters

The following table lists the parameters supported by the **create** command.

Table 1-2 Parameter description

Command	Parameter	Description

create	--annotation	Sets annotations for the container. For example, set the **native.umask** parameter.
		`--annotation native.umask=normal #The umask value of the started container is 0022.`
		`--annotation native.umask=secure #The umask value of the started container is 0027.`
		If this parameter is not set, the **umask** configuration in iSulad is used.

	--cap-drop	Deletes Linux permissions.
	--cgroup-parent	Specifies the cgroup parent path of the container.
	--cpuset-cpus	Allowed CPUs (for example, 0-3, 0, 1).
	--cpu-shares	CPU share (relative weight).
	--cpu-quota	Limits the CPU CFS quota.
	--device=[]	Adds a device to the container.
	--dns	Adds a DNS server.
	--dns-opt	Adds DNS options.
	--dns-search	Sets the search domain of a container.
	-e, --env	Sets environment variables.
	--env-file	Configures environment variables using a file.
	--entrypoint	Entry point to run when the container is started.
	--external-rootfs=PATH	Specifies a rootfs (a folder or block device) that is not managed by iSulad

		for the container.
	--files-limit	Limits the number of file handles that can be opened in a container. The value **-1** indicates no limit.
	--group-add=[]	Adds additional user groups to the container.
	--help	Displays help information.
	--health-cmd	Command executed in a container.
	--health-exit-on-unhealthy	Determines whether to kill a container when the container is detected unhealthy.
	--health-interval	Interval between two consecutive command executions.
	--health-retries	Maximum number of health check retries.
	--health-start-period	Container initialization interval.
	--health-timeout	Maximum time for executing a single check command.
	--hook-spec	Hook configuration file.
	-H, --host	Specifies the iSulad socket file path to be accessed.
	-h, --hostname	Container host name.
	-i, --interactive	Enables the standard input of the container even if it is not connected to the standard input of the container.
	--hugetlb-limit=[]	Limits the size of huge-page files, for example, **--hugetlb-limit 2MB:32MB**.
	--log-opt=[]	Log driver option. By default, the container serial port log function is disabled. You can run the **--log-opt disable-log=false** command to enable it.
	-l,--label	Sets a label for a container.
	--lablel-file	Sets container labels using files.
	-m, --memory	Memory limit.
	--memory-reservation	Sets the container memory limit. The default value is the same as that of **--memory**. **--memory** is a hard limit, and **--memory-reservation** is a soft limit. When the memory usage

		exceeds the preset value, the memory usage is dynamically adjusted (the system attempts to reduce the memory usage to a value less than the preset value when reclaiming the memory). However, the memory usage may exceed the preset value. Generally, this parameter can be used together with **--memory**. The value must be less than the preset value of **--memory**. The minimum value is 4 MB.
	--memory-swap	Memory swap space, which should be a positive integer. The value **-1** indicates no limit.
	--memory-swappiness	The value of swappiness is a positive integer ranging from 0 to 100. The smaller the value is, the less the swap partition is used and the more the memory is used in the Linux system. The larger the value is, the more the swap space is used by the kernel. The default value is **–1**, indicating that the default system value is used.
	--mount	Mounts a host directory to a container.
	--no-healthcheck	Disables the health check configuration.
	--name=NAME	Container name.
	--net=none	Connects a container to a network.
	--pids-limit	Limits the number of processes that can be executed in the container. The value **-1** indicates no limit.
	--privileged	Grants container extension privileges.
	-R, --runtime	Container runtime. The parameter value can be **lcr**, which is case insensitive. Therefore, **LCR** and **lcr** are equivalent.
	--read-only	Sets the rootfs of a container to read-only.
	--restart	Restart policy upon container exit. For a system container, **--restart on-reboot** is supported.
	--storage-opt	Configures the storage driver option for a container.

	-t, --tty	Allocates a pseudo terminal.
	--ulimit	Sets the ulimit for a container.
	-u, --user	User name or UID, in the format of **[<name|uid>][:<group|gid>]**.
	-v, --volume=[]	Mounts a volume.

Constraints

- When the **--user** or **--group-add** parameter is used to verify the user or group during container startup, if the container uses an OCI image, the verification is performed in the **etc/passwd** and **etc/group** files of the actual rootfs of the image. If a folder or block device is used as the rootfs of the container, the **etc/passwd** and **etc/group** files in the host are verified. The rootfs ignores mounting parameters such as **-v** and **--mount**. That is, when these parameters are used to attempt to overwrite the **etc/passwd** and **etc/group** files, the parameters do not take effect during the search and take effect only when the container is started. The generated configuration is saved in the **iSulad root directory/engine/container ID/start_generate_config.json** file. The file format is as follows:

```
{
    "uid": 0,
    "gid": 8,
    "additionalGids": [
        1234,
        8
    ]
}
```

Example

Create a container.

```
$ isula create busybox
fd7376591a9c3d8ee9a14f5d2c2e5255b02cc44cddaabca82170efd4497510e1
$ isula ps -a
STATUS PID IMAGE   COMMAND EXIT CODE RESTART COUNT STARTAT FINISHAT RUNTIME ID
NAMES                                                inited -   busybox "sh"    0
0           -      -       lcr     fd7376591a9c fd7376591a9c4521...
```

1.3.1.2 Starting a Container

Description

To start one or more containers, run the **isula start** command.

Usage

```
isula start [OPTIONS] CONTAINER [CONTAINER...]
```

Parameters

The following table lists the parameters supported by the **start** command.

Table 1-3 Parameter description

Command	Parameter	Description
start	-H, --host	Specifies the iSulad socket file path to be accessed.
	-R, --runtime	Container runtime. The parameter value can be **lcr**, which is case insensitive. Therefore, **LCR** and **lcr** are equivalent.

Example

Start a new container.

```
$ isula start fd7376591a9c3d8ee9a14f5d2c2e5255b02cc44cddaabca82170efd4497510e1
```

1.3.1.3 Running a Container

Description

To create and start a container, run the **isula run** command. You can use a specified container image to create a container read/write layer and prepare for running the specified command. After the container is created, run the specified command to start the container. The **run** command is equivalent to creating and starting a container.

Usage

```
isula run [OPTIONS] ROOTFS|IMAGE [COMMAND] [ARG...]
```

Parameters

The following table lists the parameters supported by the **run** command.

Table 1-4 Parameter description

Command	Parameter	Description

run	--annotation	Sets annotations for the container. For example, set the **native.umask** option.
		`--annotation native.umask=normal #The umask value of the started container is 0022.`
		`--annotation native.umask=secure #The umask value of the started container is 0027.`
		If this parameter is not set, the **umask**

		configuration in iSulad is used.
	--cap-add	Adds Linux functions.
	--cap-drop	Deletes Linux functions.
	--cgroup-parent	Specifies the cgroup parent path of the container.
	--cpuset-cpus	Allowed CPUs (for example, 0-3, 0, 1).
	--cpu-shares	CPU share (relative weight).
	--cpu-quota	Limits the CPU CFS quota.
	-d, --detach	Runs the container in the background and displays the container ID.
	--device=[]	Adds a device to the container.
	--dns	Adds a DNS server.
	--dns-opt	Adds DNS options.
	--dns-search	Sets the search domain of a container.
	-e, --env	Sets environment variables.
	--env-file	Configures environment variables using a file.
	--entrypoint	Entry point to run when the container is started.
	--external-rootfs=PATH	Specifies a rootfs (a folder or block device) that is not managed by iSulad for the container.
	--files-limit	Limits the number of file handles that can be opened in the container. The value **-1** indicates no limit.
	--group-add=[]	Adds additional user groups to the container.
	--help	Displays help information.
	--health-cmd	Command executed in a container.
	--health-exit-on-unhealthy	Determines whether to kill a container when the container is detected unhealthy.
	--health-interval	Interval between two consecutive command executions.
	--health-retries	Maximum number of health check retries.
	--health-start-period	Container initialization interval.

	--health-timeout	Maximum time for executing a single check command.
	--hook-spec	Hook configuration file.
	-H, --host	Specifies the iSulad socket file path to be accessed.
	-h, --hostname	Container host name.
	--hugetlb-limit=[]	Limits the size of huge-page files, for example, **--hugetlb-limit 2MB:32MB**.
	-i, --interactive	Enables the standard input of the container even if it is not connected to the standard input of the container.
	--log-opt=[]	Log driver option. By default, the container serial port log function is disabled. You can run the **--log-opt disable-log=false** command to enable it.
	-m, --memory	Memory limit.
	--memory-reservation	Sets the container memory limit. The default value is the same as that of **--memory**. **--memory** is a hard limit, and **--memory-reservation** is a soft limit. When the memory usage exceeds the preset value, the memory usage is dynamically adjusted (the system attempts to reduce the memory usage to a value less than the preset value when reclaiming the memory). However, the memory usage may exceed the preset value. Generally, this parameter can be used together with **--memory**. The value must be less than the preset value of **--memory**. The minimum value is 4 MB.
	--memory-swap	Memory swap space, which should be a positive integer. The value **-1** indicates no limit.
	--memory-swappiness	The value of swappiness is a positive integer ranging from 0 to 100. The smaller the value is, the less the swap partition is used and the more the memory is used in the Linux system. The larger the value is, the more the swap space is used by the kernel. The default value is **–1**, indicating that the default system value is used.

	--mount	Mounts a host directory to a container.
	--no-healthcheck	Disables the health check configuration.
	--name=NAME	Container name.
	--net=none	Connects a container to a network.
	--pids-limit	Limits the number of processes that can be executed in the container. The value **-1** indicates no limit.
	--privileged	Grants container extension privileges.
	-R, --runtime	Container runtime. The parameter value can be **lcr**, which is case insensitive. Therefore, **LCR** and **lcr** are equivalent.
	--read-only	Sets the rootfs of a container to read-only.
	--restart	Restart policy upon container exit. For a system container, **--restart on-reboot** is supported.
	--rm	Automatically clears a container upon exit.
	--storage-opt	Configures the storage driver option for a container.
	-t, --tty	Allocates a pseudo terminal.
	--ulimit	Sets the ulimit for a container.
	-u, --user	User name or UID, in the format of **[<name|uid>][:<group|gid>]**.
	-v, --volume=[]	Mounts a volume.

Constraints

- When the parent process of a container exits, the corresponding container automatically exits.
- When a common container is created, the parent process cannot be initiated because the permission of common containers is insufficient. As a result, the container does not respond when you run the **attach** command though it is created successfully.
- If **--net** is not specified when the container is running, the default host name is **localhost**.
- If the **--files-limit** parameter is to transfer a small value, for example, 1, when the container is started, iSulad creates a cgroup, sets the files.limit value, and writes the PID of the container process to the **cgroup.procs** file of the cgroup. At this time, the

container process has opened more than one handle. As a result, a write error is reported, and the container fails to be started.

- If both **--mount** and **--volume** exist and their destination paths conflict, **--mount** will be run after **--volume** (that is, the mount point in **--volume** will be overwritten).

  Note: The value of the **type** parameter of lightweight containers can be **bind** or **squashfs**. When **type** is set to **squashfs**, **src** is the image path. The value of the **type** parameter of the native Docker can be **bind**, **volume**, and **tmpfs**.

- The restart policy does not support **unless-stopped**.
- The values returned for Docker and lightweight containers are 127 and 125 respectively in the following three scenarios:

  The host device specified by **--device** does not exist.

  The hook JSON file specified by **--hook-spec** does not exist.

  The entry point specified by **--entrypoint** does not exist.

- When the **--volume** parameter is used, /dev/ptmx will be deleted and recreated during container startup. Therefore, do not mount the **/dev** directory to that of the container. Use **--device** to mount the devices in **/dev** of the container.
- Do not use the echo option to input data to the standard input of the **run** command. Otherwise, the client will be suspended. The echo value should be directly transferred to the container as a command line parameter.

```
[root@localhost ~]# echo ls | isula run -i busybox /bin/sh



^C
[root@localhost ~]#
```

  The client is suspended when the preceding command is executed because the preceding command is equivalent to input **ls** to **stdin**. Then EOF is read and the client does not send data and waits for the server to exit. However, the server cannot determine whether the client needs to continue sending data. As a result, the server is suspended in reading data, and both parties are suspended.

  The correct execution method is as follows:

```
[root@localhost ~]# isula run -i busybox ls
bin
dev
etc
home
proc
root
sys
tmp
usr
var
[root@localhost ~]#
```

- If the root directory (/) of the host is used as the file system of the container, the following situations may occur during the mounting:

Table 1-5 Mounting scenarios

Host Path (Source)	Container Path (Destination)
/home/test1	/mnt/

Host Path (Source)	Container Path (Destination)
/home/test2	/mnt/abc

NOTICE

Scenario 1: Mount **/home/test1** and then **/home/test2**. In this case, the content in **/home/test1** overwrites the content in **/mnt**. As a result, the **abc** directory does not exist in **/mnt**, and mounting **/home/test2** to **/mnt/abc** fails.

Scenario 2: Mount **/home/test2** and then **/home/test1**. In this case, the content of **/mnt** is replaced with the content of **/home/test1** during the second mounting. In this way, the content mounted during the first mounting from **/home/test2** to **/mnt/abc** is overwritten.

The first scenario is not supported. For the second scenario, users need to understand the risk of data access failures.

NOTICE

- In high concurrency scenarios (200 containers are concurrently started), the memory management mechanism of Glibc may cause memory holes and large virtual memory (for example, 10 GB). This problem is caused by the restriction of the Glibc memory management mechanism in the high concurrency scenario, but not by memory leakage. Therefore, the memory consumption does not increase infinitely. You can set the **MALLOC_ARENA_MAX** environment variable to reduce the virtual memory and increase the probability of reducing the physical memory. However, this environment variable will cause the iSulad concurrency performance to deteriorate. Set this environment variable based on the site requirements.

  To balance performance and memory usage, set **MALLOC_ARENA_MAX** to 4. (The iSulad performance deterioration on the ARM64 server is controlled by less than 10%.)

  Configuration method:
  1. To manually start iSulad, run the **export MALLOC_ARENA_MAX=4** command and then start the iSulad.
  2. If systemd manages iSulad, you can modify the **/etc/sysconfig/iSulad** file by adding **MALLOC_ARENA_MAX=4**.

Example

Run a new container.

```
$ isula run -itd busybox
9c2c13b6c35f132f49fb7ffad24f9e673a07b7fe9918f97c0591f0d7014c713b
```

1.3.1.4 Stopping a Container

Description

To stop a container, run the **isula stop** command. The SIGTERM signal is sent to the first process in the container. If the container is not stopped within the specified time (10s by default), the SIGKILL signal is sent.

Usage

```
isula stop [OPTIONS] CONTAINER [CONTAINER...]
```

Parameters

The following table lists the parameters supported by the **stop** command.

Table 1-6 Parameter description

Command	Parameter	**Description**
stop	-f, --force	Forcibly stops a running container.
	-H, --host	Specifies the iSulad socket file path to be accessed.
	-t, --time	Time for graceful stop. If the time exceeds the value of this parameter, the container is forcibly stopped.

Constraints

- If the **t** parameter is specified and the value of **t** is less than 0, ensure that the application in the container can process the stop signal.

  Principle of the Stop command: Send the SIGTERM signal to the container, and then wait for a period of time (**t** entered by the user). If the container is still running after the period of time, the SIGKILL signal is sent to forcibly kill the container.
- The meaning of the input parameter **t** is as follows:

  t < 0: Wait for graceful stop. This setting is preferred when users are assured that their applications have a proper stop signal processing mechanism.

  t = 0: Do not wait and send **kill -9** to the container immediately.

  t > 0: Wait for a specified period and send **kill -9** to the container if the container does not stop within the specified period.

  Therefore, if **t** is set to a value less than 0 (for example, **t** = -1), ensure that the container application correctly processes the SIGTERM signal. If the container ignores this signal, the container will be suspended when the **isula stop** command is run.

Example

Stop a container.

```
$ isula stop fd7376591a9c3d8ee9a14f5d2c2e5255b02cc44cddaabca82170efd4497510e1
fd7376591a9c3d8ee9a14f5d2c2e5255b02cc44cddaabca82170efd4497510e1
```

1.3.1.5 Forcibly Stopping a Container

Description

To forcibly stop one or more running containers, run the **isula kill** command.

Usage

```
isula kill [OPTIONS] CONTAINER [CONTAINER...]
```

Parameters

The following table lists the parameters supported by the **kill** command.

Table 1-7 Parameter description

Command	Parameter	Description
kill	-H, --host	Specifies the iSulad socket file path to be accessed.
	-s, --signal	Signal sent to the container.

Example

Kill a container.

```
$ isula kill fd7376591a9c3d8ee9a14f5d2c2e5255b02cc44cddaabca82170efd4497510e1
fd7376591a9c3d8ee9a14f5d2c2e5255b02cc44cddaabca82170efd4497510e1
```

1.3.1.6 Removing a Container

Description

To remove a container, run the **isula rm** command.

Usage

```
isula rm [OPTIONS] CONTAINER [CONTAINER...]
```

Parameters

The following table lists the parameters supported by the **rm** command.

Table 1-8 Parameter description

Command	Parameter	Description
rm	-f, --force	Forcibly removes a running container.
	-H, --host	Specifies the iSulad socket file path to be accessed.
	-v, --volume	Removes a volume mounted to a container. (Note: Currently, iSulad does not use this function.)

Constraints

- In normal I/O scenarios, it takes T1 to delete a running container in an empty environment (with only one container). In an environment with 200 containers (without a large number of I/O operations and with normal host I/O), it takes T2 to delete a running container. The specification of T2 is as follows: T2 = max {T1 x 3, 5}s.

Example

Delete a stopped container.

```
$ isula rm fd7376591a9c3d8ee9a14f5d2c2e5255b02cc44cddaabca82170efd4497510e1
fd7376591a9c3d8ee9a14f5d2c2e5255b02cc44cddaabca82170efd4497510e1
```

1.3.1.7 Attaching to a Container

Description

To attach standard input, standard output, and standard error of the current terminal to a running container, run the **isula attach** command. Only containers whose runtime is of the LCR type are supported.

Usage

```
isula attach [OPTIONS] CONTAINER
```

Parameters

The following table lists the parameters supported by the **attach** command.

Table 1-9 Parameter description

Command	Parameter	Description
attach	--help	Displays help information.
	-H, --host	Specifies the iSulad socket file path to be accessed.
	-D, --debug	Enables the debug mode.

Constraints

- For the native Docker, running the **attach** command will directly enter the container. For the iSulad container, you have to run the **attach** command and press **Enter** to enter the container.

Example

Attach to a running container.

```
$ isula attach fd7376591a9c3d8ee9a14f5d2c2e5255b02cc44cddaabca82170efd4497510e1
/ #
/ #
```

1.3.1.8 Renaming a Container

Description

To rename a container, run the **isula rename** command.

Usage

```
isula rename [OPTIONS] OLD_NAME NEW_NAME
```

Parameters

The following table lists the parameters supported by the **rename** command.

Table 1-10 Parameter description

Command	Parameter	**Description**
rename	-H, --host	Renames a container.

Example

Rename a container.

```
$ isula rename my_container my_new_container
```

1.3.1.9 Executing a Command in a Running Container

Description

To execute a command in a running container, run the **isula exec** command. This command is executed in the default directory of the container. If a user-defined directory is specified for the basic image, the user-defined directory is used.

Usage

```
isula exec [OPTIONS] CONTAINER COMMAND [ARG...]
```

Parameters

The following table lists the parameters supported by the **exec** command.

Table 1-11 Parameter description

Command	Parameter	**Description**
exec	-d, --detach	Runs a command in the background.
	-e, --env	Sets environment variables. (Note: Currently, iSulad does not use this function.)
	-H, --host	Specifies the iSulad socket file path

		to be accessed.
	-i, --interactive	Enables the standard input though no connection is set up. (Note: Currently, iSulad does not use this function.)
	-t, --tty	Allocates a pseudo terminal. (Note: Currently, iSulad does not use this function.)
	-u, --user	Logs in to the container as a specified user.

Constraints

- If no parameter is specified in the **isula exec** command, the **-it** parameter is used by default, indicating that a pseudo terminal is allocated and the container is accessed in interactive mode.
- When you run the **isula exec** command to execute a script and run a background process in the script, you need to use the **nohup** flag to ignore the **SIGHUP** signal.

  When you run the **isula exec** command to execute a script and run a background process in the script, you need to use the **nohup** flag. Otherwise, the kernel sends the **SIGHUP** signal to the process executed in the background when the process (first process of the session) exits. As a result, the background process exits and zombie processes occur.
- After running the **isula exec** command to access the container process, do not run background programs. Otherwise, the system will be suspended.

  To run the **isula exec** command to execute a background process, perform the following steps:

  a. Run the **isula exec container_name bash** command to access the container.

  b. After entering the container, run the **script &** command.

  c. Run the **exit** command. The terminal stops responding.

  ```
  After the isula exec command is executed to enter the container, the background program stops responding because the isula exec command is executed to enter the container and run the background while1 program. When the bash command is run to exit the process, the while1 program does not exit and becomes an orphan process, which is taken over by process 1.
  The while1 process is executed by the initial bash process fork &exec of the container. The while1 process copies the file handle of the bash process. As a result, the handle is not completely closed when the bash process exits.
  The console process cannot receive the handle closing event, epoll wait stops responding, and the process does not exit.
  ```
- Do not run the **isula exec** command in the background. Otherwise, the system may be suspended.

  Run the **isula exec** command in the background as follows:

  Run the **isula exec script &** command in the background, for example, **isula exec container_name script &,isula exec**. The command is executed in the background. The script continuously displays a file by running the **cat** command. Normally, there is output on the current terminal. If you press **Enter** on the current terminal, the client exits the stdout read operation due to the I/O read failure. As a result, the terminal does not output

data. The server continues to write data to the buffer of the FIFO because the process is still displaying files by running the **cat** command. When the buffer is full, the process in the container is suspended in the write operation.

- When a lightweight container uses the **exec** command to execute commands with pipe operations, you are advised to run the **/bin/bash -c** command.

  Typical application scenarios:

  Run the **isula exec container_name -it ls /test | grep "xx" | wc -l** command to count the number of xx files in the test directory. The output is processed by **grep** and **wc** through the pipe because **ls /test** is executed with **exec**. The output of **ls /test** executed by **exec** contains line breaks. When the output is processed, the result is incorrect.

  Cause: Run the **ls /test** command using **exec**. The command output contains a line feed character. Run the **| grep "xx" | wc -l** command for the output. The processing result is 2 (two lines).

  ```
  [root@localhost ~]# isula exec  -it container ls /test
  xx    xx10  xx12  xx14  xx3   xx5   xx7   xx9
  xx1   xx11  xx13  xx2   xx4   xx6   xx8
  [root@localhost ~]#
  ```

  Suggestion: When running the **run/exec** command to perform pipe operations, run the **/bin/bash -c** command to perform pipe operations in the container.

  ```
  [root@localhost ~]# isula exec  -it container  /bin/sh -c "ls /test | grep "xx" |
  wc -l"
  15
  [root@localhost ~]#
  ```

- Do not use the **echo** option to input data to the standard input of the **exec** command. Otherwise, the client will be suspended. The echo value should be directly transferred to the container as a command line parameter.

  ```
  [root@localhost ~]# echo ls | isula exec 38 /bin/sh



  ^C
  [root@localhost ~]#
  ```

  The client is suspended when the preceding command is executed because the preceding command is equivalent to input **ls** to **stdin**. Then EOF is read and the client does not send data and waits for the server to exit. However, the server cannot determine whether the client needs to continue sending data. As a result, the server is suspended in reading data, and both parties are suspended.

  The correct execution method is as follows:

  ```
  [root@localhost ~]# isula exec 38 ls
  bin   dev   etc   home  proc  root  sys   tmp   usr   var
  ```

Example

Run the echo command in a running container.

```
$ isula exec c75284634bee echo "hello,world"
hello,world
```

1.3.1.10 Querying Information About a Single Container

Description

To query information about a single container, run the **isula inspect** command.

Usage

```
isula inspect [OPTIONS] CONTAINER|IMAGE [CONTAINER|IMAGE...]
```

Parameters

The following table lists the parameters supported by the **inspect** command.

Table 1-12 Parameter description

Command	Parameter	**Description**
inspect	-H, --host	Specifies the iSulad socket file path to be accessed.
	-f, --format	Output format.
	-t, --time	Timeout interval, in seconds. If the **inspect** command fails to query container information within the specified period, the system stops waiting and reports an error immediately. The default value is 120s. If the value is less than or equal to 0, the **inspect** command keeps waiting until the container information is obtained successfully.

Constraints

- Lightweight containers do not support the output in {{.State}} format but support the output in the {{json .State}} format. The **-f** parameter is not supported when the object is an image.

Example

Query information about a container.

```
$ isula inspect c75284634bee
[
    {
        "Id": "c75284634beeede3ab86c828790b439d16b6ed8a537550456b1f94eb852c1c0a",
        "Created": "2019-08-01T22:48:13.993304927-04:00",
        "Path": "sh",
        "Args": [],
        "State": {
            "Status": "running",
            "Running": true,
```

```
        "Paused": false,
        "Restarting": false,
        "Pid": 21164,
        "ExitCode": 0,
        "Error": "",
        "StartedAt": "2019-08-02T06:09:25.535049168-04:00",
        "FinishedAt": "2019-08-02T04:28:09.479766839-04:00",
        "Health": {
            "Status": "",
            "FailingStreak": 0,
            "Log": []
        }
    },
    "Image": "busybox",
    "ResolvConfPath": "",
    "HostnamePath": "",
    "HostsPath": "",
    "LogPath": "none",
    "Name": "c75284634beeede3ab86c828790b439d16b6ed8a537550456b1f94eb852c1c0a",
    "RestartCount": 0,
    "HostConfig": {
        "Binds": [],
        "NetworkMode": "",
        "GroupAdd": [],
        "IpcMode": "",
        "PidMode": "",
        "Privileged": false,
        "SystemContainer": false,
        "NsChangeFiles": [],
        "UserRemap": "",
        "ShmSize": 67108864,
        "AutoRemove": false,
        "AutoRemoveBak": false,
        "ReadonlyRootfs": false,
        "UTSMode": "",
        "UsernsMode": "",
        "Sysctls": {},
        "Runtime": "lcr",
        "RestartPolicy": {
            "Name": "no",
            "MaximumRetryCount": 0
        },
        "CapAdd": [],
        "CapDrop": [],
        "Dns": [],
        "DnsOptions": [],
        "DnsSearch": [],
        "ExtraHosts": [],
        "HookSpec": "",
        "CPUShares": 0,
        "Memory": 0,
        "OomScoreAdj": 0,
        "BlkioWeight": 0,
        "BlkioWeightDevice": [],
        "CPUPeriod": 0,
```

```
            "CPUQuota": 0,
            "CPURealtimePeriod": 0,
            "CPURealtimeRuntime": 0,
            "CpusetCpus": "",
            "CpusetMems": "",
            "SecurityOpt": [],
            "StorageOpt": {},
            "KernelMemory": 0,
            "MemoryReservation": 0,
            "MemorySwap": 0,
            "OomKillDisable": false,
            "PidsLimit": 0,
            "FilesLimit": 0,
            "Ulimits": [],
            "Hugetlbs": [],
            "HostChannel": {
                "PathOnHost": "",
                "PathInContainer": "",
                "Permissions": "",
                "Size": 0
            },
            "EnvTargetFile": "",
            "ExternalRootfs": ""
        },
        "Mounts": [],
        "Config": {
            "Hostname": "localhost",
            "User": "",
            "Env": [
                "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
                "TERM=xterm",
                "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
            ],
            "Tty": true,
            "Cmd": [
                "sh"
            ],
            "Entrypoint": [],
            "Labels": {},
            "Annotations": {
                "log.console.file": "none",
                "log.console.filerotate": "7",
                "log.console.filesize": "1MB",
                "rootfs.mount": "/var/lib/isulad/mnt/rootfs",
                "native.umask": "secure"
            },
            "HealthCheck": {
                "Test": [],
                "Interval": 0,
                "Timeout": 0,
                "StartPeriod": 0,
                "Retries": 0,
                "ExitOnUnhealthy": false
            }
        },
```

```
        "NetworkSettings": {
            "IPAddress": ""
        }
    }
]
```

1.3.1.11 Querying Information About All Containers

Description

To query information about all containers, run the **isula ps** command.

Usage

```
isula ps [OPTIONS]
```

Parameters

The following table lists the parameters supported by the **ps** command.

Table 1-13 Parameter description

Command	Parameter	**Description**
ps	-a, --all	Displays all containers.
	-H, --host	Specifies the iSulad socket file path to be accessed.
	-q, --quiet	Displays only the container name.
	-f, --filter	Adds filter criteria.
	--format	Output format.
	--no-trunc	Do not truncate the container ID.

Example

Query information about all containers.

```
$ isula ps -a

ID         IMAGE                                STATUS  PID    COMMAND EXIT CODE
RESTART COUNT STARTAT       FINISHAT    RUNTIME NAMES
e84660aa059c rnd-dockerhub.huawei.com/official/busybox running 304765 "sh"    0
0           13 minutes ago -         lcr
e84660aa059cafb0a77a4002e65cc9186949132b8e57b7f4d76aa22f28fde016
$ isula ps -a --format "table {{.ID}} {{.Image}}" --no-trunc
ID                                                                 IMAGE
e84660aa059cafb0a77a4002e65cc9186949132b8e57b7f4d76aa22f28fde016
rnd-dockerhub.huawei.com/official/busybox
```

1.3.1.12 Restarting a Container

Description

To restart one or more containers, run the **isula restart** command.

Usage

```
isula restart [OPTIONS] CONTAINER [CONTAINER...]
```

Parameters

The following table lists the parameters supported by the **restart** command.

Table 1-14 Parameter description

Command	Parameter	**Description**
restart	-H, --host	Specifies the iSulad socket file path to be accessed.
	-t, --time	Time for graceful stop. If the time exceeds the value of this parameter, the container is forcibly stopped.

Constraints

- If the **t** parameter is specified and the value of **t** is less than 0, ensure that the application in the container can process the stop signal.

  The restart command first calls the stop command to stop the container. Send the SIGTERM signal to the container, and then wait for a period of time (**t** entered by the user). If the container is still running after the period of time, the SIGKILL signal is sent to forcibly kill the container.
- The meaning of the input parameter **t** is as follows:

  t < 0: Wait for graceful stop. This setting is preferred when users are assured that their applications have a proper stop signal processing mechanism.

  t = 0: Do not wait and send **kill -9** to the container immediately.

  t > 0: Wait for a specified period and send **kill -9** to the container if the container does not stop within the specified period.

  Therefore, if **t** is set to a value less than 0 (for example, **t** = -1), ensure that the container application correctly processes the SIGTERM signal. If the container ignores this signal, the container will be suspended when the **isula stop** command is run.

Example

Restart a container.

```
$ isula restart c75284634beeede3ab86c828790b439d16b6ed8a537550456b1f94eb852c1c0a
 c75284634beeede3ab86c828790b439d16b6ed8a537550456b1f94eb852c1c0a
```

1.3.1.13 Waiting for a Container to Exit

Description

To wait for one or more containers to exit, run the **isula wait** command. Only containers whose runtime is of the LCR type are supported.

Usage

```
isula wait [OPTIONS] CONTAINER [CONTAINER...]
```

Parameters

The following table lists the parameters supported by the **wait** command.

Table 1-15 Parameter description

Command	Parameter	Description
wait	-H, --host	Specifies the iSulad socket file path to be accessed.
	/	Blocks until the container stops and displays the exit code.

Example

Wait for a single container to exit.

```
$ isula wait c75284634beeede3ab86c828790b439d16b6ed8a537550456b1f94eb852c1c0a
 137
```

1.3.1.14 Viewing Process Information in a Container

Description

To view process information in a container, run the **isula top** command. Only containers whose runtime is of the LCR type are supported.

Usage

```
isula top [OPTIONS] container [ps options]
```

Parameters

The following table lists the parameters supported by the **top** command.

Table 1-16 Parameter description

Command	Parameter	Description
top	-H, --host	Specifies the iSulad socket file path

		to be accessed.
	/	Queries the process information of a running container.

Example

Query process information in a container.

```
$ isula top 21fac8bb9ea8e0be4313c8acea765c8b4798b7d06e043bbab99fc20efa72629c
UID       PID  PPID  C STIME TTY        TIME CMD
root    22166 22163  0 23:04 pts/1    00:00:00 sh
```

1.3.1.15 Displaying Resource Usage Statistics of a Container

Description

To display resource usage statistics in real time, run the **isula stats** command. Only containers whose runtime is of the LCR type are supported.

Usage

```
isula stats [OPTIONS] [CONTAINER...]
```

Parameters

The following table lists the parameters supported by the **stats** command.

Table 1-17 Parameter description

Command	Parameter	**Description**
stats	-H, --host	Specifies the iSulad socket file path to be accessed.
	-a, --all	Displays all containers. (By default, only running containers are displayed.)
	--no-stream	Display the first result only. Only statistics in non-stream mode are displayed.

Example

Display resource usage statistics.

```
$ isula stats --no-stream
21fac8bb9ea8e0be4313c8acea765c8b4798b7d06e043bbab99fc20efa72629c
CONTAINER       CPU %     MEM USAGE / LIMIT        MEM %     BLOCK I / O
PIDS
```

```
21fac8bb9ea8    0.00      56.00 KiB / 7.45 GiB      0.00      0.00 B / 0.00 B
1
```

1.3.1.16 Obtaining Container Logs

Description

To obtain container logs, run the **isula logs** command. Only containers whose runtime is of the LCR type are supported.

Usage

```
isula logs [OPTIONS] [CONTAINER...]
```

Parameters

The following table lists the parameters supported by the **logs** command.

Table 1-18 Parameter description

Command	Parameter	**Description**
logs	-H, --host	Specifies the iSulad socket file path to be accessed.
	-f, --follow	Traces log output.
	--tail	Displays the number of log records.

Constraints

- By default, the container log function is enabled. To disable this function, run the **isula create --log-opt disable-log=true** or **isula run --log-opt disable-log=true** command.

Example

Obtain container logs.

```
$ isula logs 6a144695f5dae81e22700a8a78fac28b19f8bf40e8827568b3329c7d4f742406
hello, world
hello, world
hello, world
```

1.3.1.17 Copying Data Between a Container and a Host

Description

To copy data between a host and a container, run the **isula cp** command. Only containers whose runtime is of the LCR type are supported.

Usage

```
isula cp [OPTIONS] CONTAINER:SRC_PATH DEST_PATH
isula cp [OPTIONS] SRC_PATH CONTAINER:DEST_PATH
```

Parameters

The following table lists the parameters supported by the **cp** command.

Table 1-19 Parameter description

Command	Parameter	**Description**
cp	-H, --host	Specifies the iSulad socket file path to be accessed.

Constraints

- When iSulad copies files, note that the **/etc/hostname**, **/etc/resolv.conf**, and **/etc/hosts** files are not mounted to the host, neither the **--volume** and **--mount** parameters. Therefore, the original files in the image instead of the files in the real container are copied.

  ```
  [root@localhost tmp]# isula cp b330e9be717a:/etc/hostname /tmp/hostname
  [root@localhost tmp]# cat /tmp/hostname
  [root@localhost tmp]#
  ```

- When decompressing a file, iSulad does not check the type of the file or folder to be overwritten in the file system. Instead, iSulad directly overwrites the file or folder. Therefore, if the source is a folder, the file with the same name is forcibly overwritten as a folder. If the source file is a file, the folder with the same name will be forcibly overwritten as a file.

  ```
  [root@localhost tmp]# rm -rf /tmp/test file to dir && mkdir /tmp/test file to dir
  [root@localhost tmp]# isula exec b330e9be717a /bin/sh -c "rm -rf
  /tmp/test file to dir && touch /tmp/test file to dir"
  [root@localhost tmp]# isula cp b330e9be717a:/tmp/test file to dir /tmp
  [root@localhost tmp]# ls -al /tmp | grep test file to dir
  -rw-r-----   1 root    root           0 Apr 26 09:59 test_file_to_dir
  ```

- iSulad freezes the container during the copy process and restores the container after the copy is complete.

Example

Copy the **/test/host** directory on the host to the **/test** directory on container 21fac8bb9ea8.

```
isula cp /test/host 21fac8bb9ea8:/test
```

Copy the **/www** directory on container 21fac8bb9ea8 to the **/tmp** directory on the host.

```
isula cp 21fac8bb9ea8:/www /tmp/
```

1.3.1.18 Pausing a Container

Description

To pause all processes in a container, run the **isula pause** command. Only containers whose runtime is of the LCR type are supported.

Usage

```
isula pause CONTAINER [CONTAINER...]
```

Parameters

Command	Parameter	**Description**
pause	-H, --host	Specifies the iSulad socket file path to be accessed.

Constraints

- Only containers in the running state can be paused.
- After a container is paused, other lifecycle management operations (such as **restart**, **exec**, **attach**, **kill**, **stop**, and **rm**) cannot be performed.
- After a container with health check configurations is paused, the container status changes to unhealthy.

Example

Pause a running container.

```
$ isula pause 8fe25506fb5883b74c2457f453a960d1ae27a24ee45cdd78fb7426d2022a8bac
 8fe25506fb5883b74c2457f453a960d1ae27a24ee45cdd78fb7426d2022a8bac
```

1.3.1.19 Resuming a Container

Description

To resume all processes in a container, run the **isula unpause** command. It is the reverse process of **isula pause**. Only containers whose runtime is of the LCR type are supported.

Usage

```
isula unpause CONTAINER [CONTAINER...]
```

Parameters

Command	Parameter	**Description**
pause	-H, --host	Specifies the iSulad socket file path to be accessed.

Constraints

- Only containers in the paused state can be unpaused.

Example

Resume a paused container.

```
$ isula unpause 8fe25506fb5883b74c2457f453a960d1ae27a24ee45cdd78fb7426d2022a8bac
 8fe25506fb5883b74c2457f453a960d1ae27a24ee45cdd78fb7426d2022a8bac
```

1.3.1.20 Obtaining Event Messages from the Server in Real Time

Description

The **isula events** command is used to obtain event messages such as container image lifecycle and running event from the server in real time. Only containers whose runtime type is **lcr** are supported.

Usage

```
isula events [OPTIONS]
```

Parameter

Command	Parameter	Description
events	-H, --host	Specifies the iSulad socket file path to be accessed.
	-n, --name	Obtains event messages of a specified container.
	-S, --since	Obtains event messages generated since a specified time.

Example

Run the following command to obtain event messages from the server in real time:

```
$ isula events
```

1.3.2 Interconnection with the CNI Network

1.3.2.1 Overview

The container runtime interface (CRI) is provided to connect to the CNI network, including parsing the CNI network configuration file and adding or removing a pod to or from the CNI network. When a pod needs to support a network through a container network plug-in such as Canal, the CRI needs to be interconnected to Canal so as to provide the network capability for the pod.

1.3.2.2 Common CNIs

Common CNIs include CNI network configuration items in the CNI network configuration and pod configuration. These CNIs are visible to users.

- CNI network configuration items in the CNI network configuration refer to those used to specify the path of the CNI network configuration file, path of the binary file of the CNI network plug-in, and network mode. For details, see Table 1-20.
- CNI network configuration items in the pod configuration refer to those used to set the additional CNI network list to which the pod is added. By default, the pod is added only to the default CNI network plane. You can add the pod to multiple CNI network planes as required.

Table 1-20 CNI network configuration items

Function	Command	Configuration File	Description
Path of the binary file of the CNI network plug-in	--cni-bin-dir	"cni-bin-dir": "",	The default value is **/opt/cni/bin**.
Path of the CNI network configuration file	--cni-conf-dir	"cni-conf-dir": "",	The system traverses all files with the extension .conf, .conflist, or .json in the directory. The default value is **/etc/cni/net.d**.
Network mode	--network-plugin	"network-plugin": "",	Specifies a network plug-in. The value is a null character by default, indicating that no network configuration is available and the created sandbox has only the loop NIC. The CNI and null characters are supported. Other invalid values will cause iSulad startup failure.

Additional CNI network configuration mode:

Add the network plane configuration item "network.alpha.kubernetes.io/network" to annotations in the pod configuration file.

The network plane is configured in JSON format, including:

- **name**: specifies the name of the CNI network plane.
- **interface**: specifies the name of a network interface.

The following is an example of the CNI network configuration method:

```
"annotations" : {
      "network.alpha.kubernetes.io/network": "{\"name\": \"mynet\", \"interface\":
\"eth1\"}"
 }
```

1.3.2.2.1 CNI Network Configuration Description

The CNI network configuration includes two types, both of which are in the .json file format.

- Single-network plane configuration file with the file name extension .conf or .json. For details about the configuration items, see Table 1-28 in the appendix.
- Multi-network plane configuration file with the file name extension .conflist. For details about the configuration items, see Table 1-30 in the appendix.

1.3.2.2.2 Adding a Pod to the CNI Network List

If **--network-plugin=cni** is configured for iSulad and the default network plane is configured, a pod is automatically added to the default network plane when the pod is started. If the additional network configuration is configured in the pod configuration, the pod is added to these additional network planes when the pod is started.

port_mappings in the pod configuration is also a network configuration item, which is used to set the port mapping of the pod. To set port mapping, perform the following steps:

```
"port mappings":[
    {
       "protocol": 1,
       "container port": 80,
       "host port": 8080
     }
]
```

- **protocol**: protocol used for mapping. The value can be **tcp** (identified by 0) or **udp** (identified by 1).
- **container_port**: port through which the container is mapped.
- **host_port**: port mapped to the host.

1.3.2.2.3 Removing a Pod from the CNI Network List

When StopPodSandbox is called, the interface for removing a pod from the CNI network list will be called to clear network resources.

NOTE

1. Before calling the RemovePodSandbox interface, you must call the StopPodSandbox interface at least once.
2. If StopPodSandbox fails to call the CNI, residual network resources may exist.

1.3.2.3 Usage Restrictions

- Currently, only CNI 0.3.0 and CNI 0.3.1 are supported. In later versions, CNI 0.1.0 and CNI 0.2.0 may need to be supported. Therefore, when error logs are displayed, the information about CNI 0.1.0 and CNI 0.2.0 is reserved.
- name: The value must contain lowercase letters, digits, hyphens (-), and periods (.) and cannot be started or ended with a hyphen or period. The value can contain a maximum of 200 characters.
- The number of configuration files cannot exceed 200, and the size of a single configuration file cannot exceed 1 MB.
- The extended parameters need to be configured based on the actual network requirements. Optional parameters do not need to be written into the netconf.json file.

1.3.3 Container Resource Management

1.3.3.1 Sharing Resources

Description

Containers or containers and hosts can share namespace information mutually, including PID, network, IPC, and UTS information.

Usage

When running the **isula create/run** command, you can set the namespace parameters to share resources. For details, see the following parameter description table.

Parameters

You can specify the following parameters when running the **lcrc create/run** command:

Parameter	Description	Value Range	Mandatory or Not
--pid	Specifies the PID namespace to be shared.	**[none, host, container:<containerID>]**: **none** indicates that the namespace is not shared. **host** indicates that the namespace is shared with the host. **container:<containerID>** indicates that the namespace is shared with the container.	No
--net	Specifies the network namespace to be shared.	**[none, host, container:<containerID>]**: **none** indicates that the namespace is not shared. **host** indicates that the namespace is shared with the host. **container:<containerID>** indicates that the namespace is shared with the container.	No
--ipc	Specifies the IPC namespace to be shared.	**[none, host, container:<containerID>]**: **none** indicates that the namespace is not shared. **host** indicates that the namespace is shared	No

Parameter	Description	Value Range	Mandatory or Not
		with the host. **container:<containerID>** indicates that the namespace is shared with the container.	
--uts	Specifies the UTS namespace to be shared.	**[none, host, container:<containerID>]**: **none** indicates that the namespace is not shared. **host** indicates that the namespace is shared with the host. **container:<containerID>** indicates that the namespace is shared with the container.	No

Example

If two containers need to share the same PID namespace, add **--pid container:<containerID>** when running the container. For example:

```
isula run -tid --name test pid busybox sh
isula run -tid --name test --pid container:test_pid busybox sh
```

1.3.3.2 Restricting CPU Resources of a Running Container

Description

You can set parameters to restrict the CPU resources of a container.

Usage

When running the **isula create/run** command, you can set CPU-related parameters to limit the CPU resources of a container. For details about the parameters and values, see the following table.

Parameters

You can specify the following parameters when running the **isula create/run** command:

Parameter	Description	Value Range	Mandatory or Not
--cpu-period	Limits the CPU CFS period in a container.	64-bit integer	No
--cpu-quota	Limits the CPU CFS	64-bit integer	No

Parameter	Description	Value Range	Mandatory or Not
	quota.		
--cpu-shares	Limits the CPU share (relative weight).	64-bit integer	No
--cpuset-cpus	Limits the CPU nodes.	A character string. The value is the number of CPUs to be configured. The value ranges from 0 to 3, or 0 and 1.	No
--cpuset-mems	Limits the memory nodes used by cpuset in the container.	A character string. The value is the number of CPUs to be configured. The value ranges from 0 to 3, or 0 and 1.	No

Example

To restrict a container to use a specific CPU, add **--cpuset-cpus number** when running the container. For example:

```
isula run -tid --cpuset-cpus 0,2-3 busybox sh
```

NOTE

You can check whether the configuration is successful. For details, see "Querying Information About a Single Container."

1.3.3.3 Restricting the Memory Usage of a Running Container

Description

You can set parameters to restrict the memory usage of a container.

Usage

When running the **isula create/run** command, you can set memory-related parameters to restrict memory usage of containers. For details about the parameters and values, see the following table.

Parameters

You can specify the following parameters when running the **isula create/run** command:

Parameter	Description	Value Range	Mandatory or Not
--memory	Specifies the upper limit of the memory usage of a container.	64-bit integer The value is a non-negative number. The value **0** indicates that no	No

Parameter	Description	Value Range	Mandatory or Not
		limit is set. The unit can be empty (byte), KB, MB, GB, TB, or PB.	
--memory-reservation	Specifies the soft upper limit of the memory of a container.	64-bit integer The value is a non-negative number. The value **0** indicates that no limit is set. The unit can be empty (byte), KB, MB, GB, TB, or PB.	No
--memory-swap	Specifies the upper limit of the swap memory of the container.	64-bit integer The value can be **-1** or a non-negative number. The value **-1** indicates no limit, and the value **0** indicates that no limit is set. The unit can be empty (byte), KB, MB, GB, TB, or PB.	No
--kernel-memory	Specifies the upper limit of the kernel memory of the container.	64-bit integer The value is a non-negative number. The value **0** indicates that no limit is set. The unit can be empty (byte), KB, MB, GB, TB, or PB.	No

Example

To set the upper limit of the memory of a container, add **--memory <number>[<unit>]** when running the container. For example:

```
isula run -tid --memory 1G busybox sh
```

1.3.3.4 Restricting I/O Resources of a Running Container

Description

You can set parameters to limit the read/write speed of devices in the container.

Usage

When running the **isula create/run** command, you can set **--device-read-bps/--device-write-bps <device-path>:<number>[<unit>]** to limit the read/write speed of devices in the container.

Parameters

When running the **isula create/run** command, set **--device-read/write-bps**.

Parameter	Description	Value Range	Mandatory or Not
--device-read-bps/--device-write-bps	Limits the read/write speed of devices in the container.	64-bit integer The value is a positive integer. The value can be 0, indicating that no limit is set. The unit can be empty (byte), KB, MB, GB, TB, or PB.	No

Example

To limit the read/write speed of devices in the container, add **--device-write-bps/--device-read-bps <device-path>:<number>[<unit>]** when running the container. For example, to limit the read speed of the device **/dev/sda** in the container **busybox** to 1 Mbit/s, run the following command:

```
isula run -tid --device-write /dev/sda:1mb busybox sh
```

To limit the write speed, run the following command:

```
isula run -tid read-bps /dev/sda:1mb busybox sh
```

1.3.3.5 Restricting the Rootfs Storage Space of a Container

Description

When the overlay2 storage driver is used on the EXT4 file system, the file system quota of a single container can be set. For example, the quota of container A is set to 5 GB, and the quota of container B is set to 10 GB.

This feature is implemented by the project quota function of the EXT4 file system. If the kernel supports this function, use the syscall SYS_IOCTL to set the project ID of a directory, and then use the syscall SYS_QUOTACTL to set the hard limit and soft limit of the corresponding project ID.

Usage

1. Prepare the environment.

   Ensure that the file system supports the **Project ID** and **Project Quota** attributes, the kernel version is 4.19 or later, and the version of the peripheral package e2fsprogs is 1.43.4-2 or later.

2. Before mounting overlayfs to a container, set different project IDs for the upper and work directories of different containers and set inheritance options. After overlayfs is mounted to a container, the project IDs and inherited attributes cannot be modified.
3. Set the quota as a privileged user outside the container.
4. Add the following configuration to daemon:

   ```
   -s overlay2 --storage-opt overlay2.override_kernel_check=true
   ```

5. Daemon supports the following options for setting default restrictions for containers:

   --storage-opt overlay2.basesize=128M specifies the default limit. If **--storeage-opt size** is also specified when you run the **isula run** command, the value of this parameter takes effect. If no size is specified during the daemon process or when you run the **isula run** command, the size is not limited.

6. Enable the **Project ID** and **Project Quota** attributes of the file system.
   - Format and mount the file system.

     ```
     # mkfs.ext4 -O quota,project /dev/sdb
     # mount -o prjquota /dev/sdb /var/lib/isulad
     ```

Parameters

When running the **create/run** command, set **--storage-opt**.

Parameter	Description	Value Range	Mandatory or Not
--storage-opt size=${rootfsSize}	Restricts the root file system (rootfs) storage space of the container.	The size parsed by **rootfsSize** is a positive 64-bit integer expressed in bytes. You can also set it to **([kKmMgGtTpP])?[iI]?[bB]?$**.	No

Example

In the **isula run/create** command, use the existing parameter **--storage-opt size=***value* to set the quota. The value is a positive number in the unit of **[kKmMgGtTpP]?[iI]?[bB]?**. If the value does not contain a unit, the default unit is byte.

```
$ [root@localhost ~]# isula run -ti --storage-opt size=10M busybox
/ # df -h
Filesystem              Size     Used Available Use% Mounted on
overlay                10.0M     48.0K     10.0M   0% /
none                   64.0M         0     64.0M   0% /dev
none                   10.0M         0     10.0M   0% /sys/fs/cgroup
tmpfs                  64.0M         0     64.0M   0% /dev
shm                    64.0M         0     64.0M   0% /dev/shm
/dev/mapper/vg--data-ext41
                       9.8G     51.5M      9.2G   1% /etc/hostname
/dev/mapper/vg--data-ext41
                       9.8G     51.5M      9.2G   1% /etc/resolv.conf
/dev/mapper/vg--data-ext41
                       9.8G     51.5M      9.2G   1% /etc/hosts
```

```
tmpfs                    3.9G        0     3.9G   0% /proc/acpi
tmpfs                   64.0M        0    64.0M   0% /proc/kcore
tmpfs                   64.0M        0    64.0M   0% /proc/keys
tmpfs                   64.0M        0    64.0M   0% /proc/timer list
tmpfs                   64.0M        0    64.0M   0% /proc/sched debug
tmpfs                    3.9G        0     3.9G   0% /proc/scsi
tmpfs                   64.0M        0    64.0M   0% /proc/fdthreshold
tmpfs                   64.0M        0    64.0M   0% /proc/fdenable
tmpfs                    3.9G        0     3.9G   0% /sys/firmware
/ #
/ # dd if=/dev/zero of=/home/img bs=1M count=12 && sync
dm-4: write failed, project block limit reached.
10+0 records in
9+0 records out
10432512 bytes (9.9MB) copied, 0.011782 seconds, 844.4MB/s
/ # df -h | grep overlay
overlay               10.0M    10.0M        0 100% /
/ #
```

Constraints

1. The quota applies only to the rw layer.

   The quota of overlay2 is for the rw layer of the container. The image size is not included.

2. The kernel supports and enables this function.

   The kernel must support the EXT4 project quota function. When running **mkfs**, add **-O quota,project**. When mounting the file system, add **-o prjquota**. If any of the preceding conditions is not met, an error is reported when **--storage-opt size=***value* is used.

   ```
   $ [root@localhost ~]# isula run -it --storage-opt size=10Mb busybox df -h
   Error response from daemon: Failed to prepare rootfs with error:
   time="2019-04-09T05:13:52-04:00" level=fatal msg="error creating read-
   write layer with ID
   "a4c0e55e82c55e4ee4b0f4ee07f80cc2261cf31b2c2dfd628fa1fb00db97270f":
   --storage-opt is supported only for overlay over
   xfs or ext4 with 'pquota' mount option"
   ```

3. Description of the limit of quota:

   a. If the quota is greater than the size of the partition where user **root** of iSulad is located, the file system quota displayed by running the **df** command in the container is the size of the partition where user **root** of iSulad is located, not the specified quota.

   b. **--storage-opt size=0** indicates that the size is not limited and the value cannot be less than 4096. The precision of size is one byte. If the specified precision contains decimal bytes, the decimal part is ignored. For example, if size is set to **0.1**, the size is not limited. (The value is restricted by the precision of the floating point number stored on the computer. That is, 0.99999999999999999999999999 is equal to 1. The number of digits 9 may vary according to computers. Therefore, 4095.99999999999999999999999999 is equal to 4096.) Note that running **isula inspect** displays the original command line specified format. If the value contains decimal bytes, you need to ignore the decimal part.

   c. If the quota is too small, for example, **--storage-opt size=4k**, the container may fail to be started because some files need to be created for starting the container.

d. The **-o prjquota** option is added to the root partition of iSulad when iSulad is started last time. If this option is not added during this startup, the setting of the container with quota created during the last startup does not take effect.

e. The value range of the daemon quota **--storage-opt overlay2.basesize** is the same as that of **--storage-opt size**.

4. When **storage-opt** is set to 4 KB, the lightweight container startup is different from that of Docker.

   Use the **storage-opt size=4k** and image **rnd-dockerhub.huawei.com/official/ubuntu-arm64:latest** to run the container.

   Docker fails to be started.

```
[root@localhost ~]# docker run -itd --storage-opt size=4k
rnd-dockerhub.huawei.com/official/ubuntu-arm64:latest
docker: Error response from daemon: symlink /proc/mounts
/var/lib/docker/overlay2/e6e12701db1a488636c881b44109a807e187b8db51a50015db34a1
31294fcf70-init/merged/etc/mtab: disk quota exceeded.
See 'docker run --help'.
```

   The lightweight container is started properly and no error is reported.

```
[root@localhost ~]# isula run -itd --storage-opt size=4k
rnd-dockerhub.huawei.com/official/ubuntu-arm64:latest
636480b1fc2cf8ac895f46e77d86439fe2b359a1ff78486ae81c18d089bbd728
[root@localhost ~]# isula ps
STATUS  PID   IMAGE                                          COMMAND   EXIT CODE
RESTART COUNT STARTAT      FINISHAT RUNTIME ID         NAMES
running 17609 rnd-dockerhub.huawei.com/official/ubuntu-arm64:latest /bin/bash 0
0           2 seconds ago -       lcr    636480b1fc2c
636480b1fc2cf8ac895f46e77d86439fe2b359a1ff78486ae81c18d089bbd728
```

   During container startup, if you need to create a file in the **rootfs** directory of the container, the image size exceeds 4 KB, and the quota is set to 4 KB, the file creation will fail.

   When Docker starts the container, it creates more mount points than iSulad to mount some directories on the host to the container, such as **/proc/mounts** and **/dev/shm**. If these files do not exist in the image, the creation will fail, therefore, the container fails to be started.

   When a lightweight container uses the default configuration during container startup, there are few mount points. The lightweight container is created only when the directory like **/proc** or **/sys** does not exist. The image **rnd-dockerhub.huawei.com/official/ubuntu-arm64:latest** in the test case contains **/proc** and **/sys**. Therefore, no new file or directory is generated during the container startup. As a result, no error is reported during the lightweight container startup. To verify this process, when the image is replaced with **rnd-dockerhub.huawei.com/official/busybox-aarch64:latest**, an error is reported when the lightweight container is started because **/proc** does not exist in the image.

```
[root@localhost ~]# isula run -itd --storage-opt size=4k
rnd-dockerhub.huawei.com/official/busybox-aarch64:latest
8e893ab483310350b8caa3b29eca7cd3c94eae55b48bfc82b350b30b17a0aaf4
Error response from daemon: Start container error: runtime error:
8e893ab483310350b8caa3b29eca7cd3c94eae55b48bfc82b350b30b17a0aaf4:tools/lxc star
t.c:main:404 starting container process caused "Failed to setup lxc,
please check the config file."
```

5. Other description:

When using iSulad with the quota function to switch data disks, ensure that the data disks to be switched are mounted using the **prjquota** option and the mounting mode of the **/var/lib/isulad/storage/overlay2** directory is the same as that of the **/var/lib/isulad** directory.

NOTE

Before switching the data disk, ensure that the mount point of **/var/lib/isulad/storage/overlay2** is unmounted.

1.3.3.6 Restricting the Number of File Handles in a Container

Description

You can set parameters to limit the number of file handles that can be opened in a container.

Usage

When running the **isula create/run** command, set the **--files-limit** parameter to limit the number of file handles that can be opened in a container.

Parameters

Set the **--files-limit** parameter when running the **isula create/run** command.

Parameter	Description	Value Range	Mandatory or Not
--files-limit	Limits the number of file handles that can be opened in a container.	64-bit integer The value can be 0 or a negative number, but cannot be greater than 2 to the power of 63 minus 1. The value 0 or a negative number indicates no limit. During container creation, some handles are opened temporarily. Therefore, the value cannot be too small. Otherwise, the container may not be restricted by the file limit. If the value is less than the number of opened handles, the cgroup file cannot be written. It is recommended that the value be greater than 30.	No

Example

When running the container, add **--files-limit n**. For example:

```
isula run -ti --files-limit 1024 busybox bash
```

Constraints

1. If the **--files-limit** parameter is set to a small value, for example, 1, the container may fail to be started.

   ```
   [root@localhost ~]# isula run -itd --files-limit 1
   rnd-dockerhub.huawei.com/official/busybox-aarch64
   004858d9f9ef429b624f3d20f8ba12acfbc8a15bb121c4036de4e5745932eff4
   Error response from daemon: Start container error: Container is not
   running:004858d9f9ef429b624f3d20f8ba12acfbc8a15bb121c4036de4e5745932eff4
   ```

   Docker will be started successfully, and the value of **files.limit cgroup** is **max**.

   ```
   [root@localhost ~]# docker run -itd --files-limit 1
   rnd-dockerhub.huawei.com/official/busybox-aarch64
   ef9694bf4d8e803a1c7de5c17f5d829db409e41a530a245edc2e5367708dbbab
   [root@localhost ~]# docker exec -it ef96 cat /sys/fs/cgroup/files/files.limit
   max
   ```

   The root cause is that the startup principles of the lxc and runc processes are different. After the lxc process creates the cgroup, the files.limit value is set, and then the PID of the container process is written into the cgroup.procs file of the cgroup. At this time, the process has opened more than one handle. As a result, an error is reported, and the startup fails. After you create a cgroup by running the **runc** command, the PID of the container process is written to the cgroup.procs file of the cgroup, and then the files.limit value is set. Because more than one handle is opened by the process in the cgroup, the file.limit value does not take effect, the kernel does not report any error, and the container is started successfully.

1.3.3.7 Restricting the Number of Processes or Threads that Can Be Created in a Container

Description

You can set parameters to limit the number of processes or threads that can be created in a container.

Usage

When creating or running a container, use the **--pids-limit** parameter to limit the number of processes or threads that can be created in the container.

Parameters

When running the **create/run** command, set the **--pids-limit** parameter.

Parameter	Description	Value Range	Mandatory or Not
--pids-limit	Limits the number of file handles that can be opened in a	64-bit integer The value can be 0 or a negative number,	No

Parameter	Description	Value Range	Mandatory or Not
	container.	but cannot be greater than 2 to the power of 63 minus 1. The value 0 or a negative number indicates no limit.	

Example

When running the container, add **--pids-limit n**. For example:

```
isula run -ti --pids-limit 1024 busybox bash
```

Constraints

During container creation, some processes are created temporarily. Therefore, the value cannot be too small. Otherwise, the container may fail to be started. It is recommended that the value be greater than 10.

1.3.3.8 Configuring the ulimit Value in a Container

Description

You can use parameters to control the resources for executed programs.

Usage

Set the **--ulimit** parameter when creating or running a container, or configure the parameter on the daemon to control the resources for executed programs in the container.

Parameters

Use either of the following methods to configure ulimit:

1. When running the **isula create/run** command, use **--ulimit <type>=<soft>[:<hard>]** to control the resources of the executed shell program.

Parameter	Description	Value Range	Mandatory or Not
--ulimit	Limits the resources of the executed shell program.	64-bit integer The value of the soft limit must be less than or equal to that of the hard limit. If only the soft limit is specified, the value of the hard limit is equal to that of the soft limit. Some types of resources do not support	No

Parameter	Description	Value Range	Mandatory or Not
		negative numbers. For details, see the following table.	

2. Use daemon parameters or configuration files.

   For details, see --default-ulimits in 1.2.3.1 Deployment Mode.

   --ulimit can limit the following types of resources:

Type	Description	Value Range
core	limits the core file size (KB)	64-bit integer, without unit. The value can be 0 or a negative number. The value -1 indicates no limit. Other negative numbers are forcibly converted into a large positive integer.
cpu	max CPU time (MIN)	
data	max data size (KB)	
fsize	maximum filesize (KB)	
locks	max number of file locks the user can hold	
memlock	max locked-in-memory address space (KB)	
msgqueue	max memory used by POSIX message queues (bytes)	
nice	nice priority	
nproc	max number of processes	
rss	max resident set size (KB)	
rtprio	max realtime priority	
rttime	realtime timeout	
sigpending	max number of pending signals	
stack	max stack size (KB)	
nofile	max number of open file descriptors	64-bit integer, without unit. The value cannot be negative. A negative number is forcibly converted to a large positive number. In addition, "Operation not permitted" is displayed during the setting.

Example

When creating or running a container, add **--ulimit <type>=<soft>[:<hard>]**. For example:

```
isula create/run -tid --ulimit nofile=1024:2048 busybox sh
```

Constraints

The ulimit cannot be configured in the **daemon.json** and **/etc/sysconfig/iSulad** files (or the iSulad command line). Otherwise, an error is reported when iSulad is started.

1.3.4 Privileged Container

1.3.4.1 Scenarios

By default, iSulad starts common containers that are suitable for starting common processes. However, common containers have only the default permissions defined by capabilities in the **/etc/default/isulad/config.json** directory. To perform privileged operations (such as use devices in the **/sys** directory), a privileged container is required. By using this feature, user **root** in the container has **root** permissions of the host. Otherwise, user **root** in the container has only common user permissions of the host.

1.3.4.2 Usage Restrictions

Privileged containers provide all functions for containers and remove all restrictions enforced by the device cgroup controller. A privileged container has the following features:

- Secomp does not block any system call.
- The **/sys** and **/proc** directories are writable.
- All devices on the host can be accessed in the container.
- All system capabilities will be enabled.

Default capabilities of a common container are as follows:

Capability Key	Description
SETPCAP	Modifies the process capabilities.
MKNOD	Allows using the system call mknod() to create special files.
AUDIT_WRITE	Writes records to kernel auditing logs.
CHOWN	Modifies UIDs and GIDs of files. For details, see the chown(2).
NET_RAW	Uses RAW and PACKET sockets and binds any IP address to the transparent proxy.
DAC_OVERRIDE	Ignores the discretionary access control (DAC) restrictions on files.
FOWNER	Ignores the restriction that the file owner ID must be the same as the process user ID.
FSETID	Allows setting setuid bits of files.

Capability Key	Description
KILL	Allows sending signals to processes that do not belong to itself.
SETGID	Allows the change of the process group ID.
SETUID	Allows the change of the process user ID.
NET_BIND_SERVICE	Allows bounding to a port whose number is smaller than 1024.
SYS_CHROOT	Allows using the system call chroot().
SETFCAP	Allows transferring and deleting capabilities to other processes.

When a privileged container is enabled, the following capabilities are added:

Capability Key	Description
SYS_MODULE	Loads and unloads kernel modules.
SYS_RAWIO	Allows direct access to **/devport**, **/dev/mem**, **/dev/kmem**, and original block devices.
SYS_PACCT	Allows the process BSD audit.
SYS_ADMIN	Allows executing system management tasks, such as loading or unloading file systems and setting disk quotas.
SYS_NICE	Allows increasing the priority and setting the priorities of other processes.
SYS_RESOURCE	Ignores resource restrictions.
SYS_TIME	Allows changing the system clock.
SYS_TTY_CONFIG	Allows configuring TTY devices.
AUDIT_CONTROL	Enables and disables kernel auditing, modifies audit filter rules, and extracts audit status and filtering rules.
MAC_ADMIN	Overrides the mandatory access control (MAC), which is implemented for the Smack Linux Security Module (LSM).
MAC_OVERRIDE	Allows MAC configuration or status change, which is implemented for Smack LSM.
NET_ADMIN	Allows executing network management tasks.
SYSLOG	Performs the privileged syslog(2) operation.
DAC_READ_SEARCH	Ignores the DAC access restrictions on file reading and catalog search.

Capability Key	Description
LINUX_IMMUTABLE	Allows modifying the IMMUTABLE and APPEND attributes of a file.
NET_BROADCAST	Allows network broadcast and multicast access.
IPC_LOCK	Allows locking shared memory segments.
IPC_OWNER	Ignores the IPC ownership check.
SYS_PTRACE	Allows tracing any process.
SYS_BOOT	Allows restarting the OS.
LEASE	Allows modifying the FL_LEASE flag of a file lock.
WAKE_ALARM	Triggers the function of waking up the system, for example, sets the CLOCK_REALTIME_ALARM and CLOCK_BOOTTIME_ALARM timers.
BLOCK_SUSPEND	Allows blocking system suspension.

1.3.4.3 Usage Guide

iSulad runs the **--privileged** command to enable the privilege mode for containers. Do not add privileges to containers unless necessary. Comply with the principle of least privilege to reduce security risks.

```
isula run --rm -it --privileged busybox
```

1.3.5 CRI

1.3.5.1 Description

The Container Runtime Interface (CRI) provided by Kubernetes defines container and image service APIs. iSulad uses the CRI to interconnect with Kubernetes.

Since the container runtime is isolated from the image lifecycle, two services need to be defined. This API is defined by using Protocol Buffer based on gRPC.

The current CRI version is v1alpha1. For official API description, access the following link:

https://github.com/kubernetes/blob/release-1.14/pkg/kubelet/apis/cri/runtime/v1alpha2/api.proto

iSulad uses the API description file of version 1.14 used by Pass, which is slightly different from the official API description file. API description in this document prevails.

NOTE

The listening IP address of the CRI WebSocket streaming service is **127.0.0.1** and the port number is **10350**. The port number can be configured in the **--websocket-server-listening-port** command or in the **daemon.json** configuration file.

1.3.5.2 APIs

The following tables list the parameters that may be used in each API. Some parameters do not take effect now, which have been noted in the corresponding parameter description.

API Parameters

- **DNSConfig**

  The API is used to configure DNS servers and search domains of a sandbox.

Parameter	Description
repeated string servers	DNS server list of a cluster.
repeated string searches	DNS search domain list of a cluster.
repeated string options	DNS option list. For details, see https://linux.die.net/man/5/resolv.conf.

- **Protocol**

  The API is used to specify enum values of protocols.

Parameter	Description
TCP = 0↵	Transmission Control Protocol (TCP).
UDP = 1	User Datagram Protocol (UDP).

- **PortMapping**

  The API is used to configure the port mapping for a sandbox.

Parameter	Description
Protocol protocol	Protocol used for port mapping.
int32 container_port	Port number in the container.
int32 host_port	Port number on the host.
string host_ip	Host IP address.

- **MountPropagation**

  The API is used to specify enums of mount propagation attributes.

Parameter	Description
PROPAGATION_PRIVATE = 0	No mount propagation attributes, that is, private in Linux.
PROPAGATION_HOST_TO_CONTAINER = 1	Mount attribute that can be propagated from the host to the container, that is, rslave in Linux.
PROPAGATION_BIDIRECTIO	Mount attribute that can be propagated between a host

Parameter	Description
NAL = 2	and a container, that is, rshared in Linux.

- **Mount**

  The API is used to mount a volume on the host to a container. (Only files and folders are supported.)

Parameter	Description
string container_path	Path in the container.
string host_path	Path on the host.
bool readonly	Whether the configuration is read-only in the container. Default value: **false**
bool selinux_relabel	Whether to set the SELinux label. This parameter does not take effect now.
MountPropagation propagation	Mount propagation attribute. The value can be **0**, **1**, or **2**, corresponding to the private, rslave, and rshared propagation attributes respectively. Default value: **0**

- **NamespaceOption**

Parameter	Description
bool host_network	Whether to use host network namespaces.
bool host_pid	Whether to use host PID namespaces.
bool host_ipc	Whether to use host IPC namespaces.

- **Capability**

  This API is used to specify the capabilities to be added and deleted.

Parameter	Description
repeated string add_capabilities	Capabilities to be added.
repeated string drop_capabilities	Capabilities to be deleted.

- **Int64Value**

  The API is used to encapsulate data of the signed 64-bit integer type.

Parameter	Description

Parameter	Description
int64 value	Actual value of the signed 64-bit integer type.

- **UInt64Value**

  The API is used to encapsulate data of the unsigned 64-bit integer type.

Parameter	Description
uint64 value	Actual value of the unsigned 64-bit integer type.

- **LinuxSandboxSecurityContext**

  The API is used to configure the Linux security options of a sandbox.

  Note that these security options are not applied to containers in the sandbox, and may not be applied to the sandbox without any running process.

Parameter	Description
NamespaceOption namespace_options	Sandbox namespace options.
SELinuxOption selinux_options	SELinux options. This parameter does not take effect now.
Int64Value run_as_user	Process UID in the sandbox.
bool readonly_rootfs	Whether the root file system of the sandbox is read-only.
repeated int64 supplemental_groups	Information of the user group of the init process in the sandbox (except the primary GID).
bool privileged	Whether the sandbox is a privileged container.
string seccomp_profile_path	Path of the seccomp configuration file. Valid values are as follows: // **unconfined**: Seccomp is not configured. // **localhost/** *Full path of the configuration file*: configuration file path installed in the system. // *Full path of the configuration file*: full path of the configuration file. // **unconfined** is the default value.

- **LinuxPodSandboxConfig**

  The API is used to configure information related to the Linux host and containers.

Parameter	Description
string cgroup_parent	Parent path of the cgroup of the sandbox. The runtime can use the cgroupfs or systemd syntax

Parameter	Description
	based on site requirements. This parameter does not take effect now.
LinuxSandboxSecurityContext security_context	Security attribute of the sandbox.
map<string, string> sysctls	Linux sysctls configuration of the sandbox.

- **PodSandboxMetadata**

  Sandbox metadata contains all information that constructs a sandbox name. It is recommended that the metadata be displayed on the user interface during container running to improve user experience. For example, a unique sandbox name can be generated based on the metadata during running.

Parameter	Description
string name	Sandbox name.
string uid	Sandbox UID.
string namespace	Sandbox namespace.
uint32 attempt	Number of attempts to create a sandbox. Default value: **0**

- **PodSandboxConfig**

  This API is used to specify all mandatory and optional configurations for creating a sandbox.

Parameter	Description
PodSandboxMetadata metadata	Sandbox metadata, which uniquely identifies a sandbox. The runtime must use the information to ensure that operations are correctly performed, and to improve user experience, for example, construct a readable sandbox name.
string hostname	Host name of the sandbox.
string log_directory	Folder for storing container log files in the sandbox.
DNSConfig dns_config	Sandbox DNS configuration.
repeated PortMapping port_mappings	Sandbox port mapping.
map<string, string> labels	Key-value pair that can be used to identify a sandbox or a series of sandboxes.
map<string, string> annotations	Key-value pair that stores any information, whose values cannot be changed and can be queried by using the PodSandboxStatus API.

Parameter	Description
LinuxPodSandboxConfig linux	Options related to the Linux host.

- **PodSandboxNetworkStatus**

  The API is used to describe the network status of a sandbox.

Parameter	Description
string ip	IP address of the sandbox.
string name	Network interface name in the sandbox.
string network	Name of the additional network.

- **Namespace**

  The API is used to set namespace options.

Parameter	Description
NamespaceOption options	Linux namespace options.

- **LinuxPodSandboxStatus**

  The API is used to describe the status of a Linux sandbox.

Parameter	Description
Namespace **namespaces**	Sandbox namespace.

- **PodSandboxState**

  The API is used to specify enum data of the sandbox status values.

Parameter	Description
SANDBOX_READY = 0	The sandbox is ready.
SANDBOX_NOTREADY = 1	The sandbox is not ready.

- **PodSandboxStatus**

  The API is used to describe the PodSandbox status.

Parameter	Description
string id	Sandbox ID.
PodSandboxMetadata metadata	Sandbox metadata.
PodSandboxState state	Sandbox status value.

Parameter	Description
int64 created_at	Sandbox creation timestamp (unit: ns).
repeated PodSandboxNetworkStatus networks	Multi-plane network status of the sandbox.
LinuxPodSandboxStatus linux	Sandbox status complying with the Linux specifications.
map<string, string> labels	Key-value pair that can be used to identify a sandbox or a series of sandboxes.
map<string, string> annotations	Key-value pair that stores any information, whose values cannot be changed by the runtime.

- **PodSandboxStateValue**

  The API is used to encapsulate PodSandboxState.

Parameter	Description
PodSandboxState state	Sandbox status value.

- **PodSandboxFilter**

  The API is used to add filter criteria for the sandbox list. The intersection of multiple filter criteria is displayed.

Parameter	Description
string id	Sandbox ID.
PodSandboxStateValue state	Sandbox status.
map<string, string> label_selector	Sandbox label, which does not support regular expressions and must be fully matched.

- **PodSandbox**

  This API is used to provide a minimum description of a sandbox.

Parameter	Description
string id	Sandbox ID.
PodSandboxMetadata metadata	Sandbox metadata.
PodSandboxState state	Sandbox status value.
int64 created_at	Sandbox creation timestamp (unit: ns).
map<string, string> labels	Key-value pair that can be used to identify a sandbox or a series of sandboxes.

Parameter	Description
map<string, string> annotations	Key-value pair that stores any information, whose values cannot be changed by the runtime.

- **KeyValue**

  The API is used to encapsulate key-value pairs.

Parameter	Description
string key	Key
string value	Value

- **SELinuxOption**

  The API is used to specify the SELinux label of a container.

Parameter	Description
string user	User
string role	Role
string type	Type
string level	Level

- **ContainerMetadata**

  Container metadata contains all information that constructs a container name. It is recommended that the metadata be displayed on the user interface during container running to improve user experience. For example, a unique container name can be generated based on the metadata during running.

Parameter	Description
string name	Container name.
uint32 attempt	Number of attempts to create a container. Default value: **0**

- **ContainerState**

  The API is used to specify enums of container status values.

Parameter	Description
CONTAINER_CREATED = 0	The container is created.
CONTAINER_RUNNING = 1	The container is running.
CONTAINER_EXITED = 2	The container exits.

Parameter	Description
CONTAINER_UNKNOWN = 3	Unknown container status.

- **ContainerStateValue**

  The API is used to encapsulate the data structure of ContainerState.

Parameter	Description
ContainerState **state**	Container status value.

- **ContainerFilter**

  The API is used to add filter criteria for the container list. The intersection of multiple filter criteria is displayed.

Parameter	Description
string id	Container ID.
PodSandboxStateValue state	Container status.
string pod_sandbox_id	Sandbox ID.
map<string, string> label_selector	Container label, which does not support regular expressions and must be fully matched.

- **LinuxContainerSecurityContext**

  The API is used to specify container security configurations.

Parameter	Description
Capability capabilities	Added or removed capabilities.
bool privileged	Whether the container is in privileged mode. Default value: **false**
NamespaceOption namespace_options	Container namespace options.
SELinuxOption selinux_options	SELinux context, which is optional. This parameter does not take effect now.
Int64Value run_as_user	UID for running container processes. Only **run_as_user** or **run_as_username** can be specified at a time. **run_as_username** takes effect preferentially.
string run_as_username	Username for running container processes. If specified, the user must exist in **/etc/passwd** in the container image and be parsed by the runtime. Otherwise, an error must occur during running.

bool readonly_rootfs	Whether the root file system in a container is read-only. The default value is configured in **config.json**.
repeated int64 supplemental_groups	List of user groups of the init process running in the container (except the primary GID).
string apparmor_profile	AppArmor configuration file of the container. This parameter does not take effect now.
string seccomp_profile_path	Path of the seccomp configuration file of the container.
bool no_new_privs	Whether to set the no_new_privs flag in the container.

- **LinuxContainerResources**

  The API is used to specify configurations of Linux container resources.

Parameter	**Description**
int64 cpu_period	CPU CFS period. Default value: **0**
int64 cpu_quota	CPU CFS quota. Default value: **0**
int64 cpu_shares	CPU share (relative weight). Default value: **0**
int64 memory_limit_in_bytes	Memory limit (unit: byte). Default value: **0**
int64 oom_score_adj	OOMScoreAdj that is used to adjust the OOM killer. Default value: **0**
string cpuset_cpus	CPU core used by the container. Default value: null
string cpuset_mems	Memory nodes used by the container. Default value: null

- **Image**

  The API is used to describe the basic information about an image.

Parameter	**Description**
string id	Image ID.
repeated string repo_tags	Image tag name **repo_tags**.
repeated string repo_digests	Image digest information.
uint64 size	Image size.
Int64Value uid	Default image UID.
string username	Default image username.

- **ImageSpec**

The API is used to represent the internal data structure of an image. Currently, ImageSpec encapsulates only the container image name.

Parameter	Description
string image	Container image name.

- **StorageIdentifier**

  The API is used to specify the unique identifier for defining the storage.

Parameter	Description
string uuid	Device UUID.

- **FilesystemUsage**

Parameter	Description
int64 timestamp	Timestamp when file system information is collected.
StorageIdentifier storage_id	UUID of the file system that stores images.
UInt64Value used_bytes	Size of the metadata that stores images.
UInt64Value inodes_used	Number of inodes of the metadata that stores images.

- **AuthConfig**

Parameter	Description
string username	Username used for downloading images.
string password	Password used for downloading images.
string auth	Authentication information used for downloading images. The value is encoded by using Base64.
string server_address	IP address of the server where images are downloaded. This parameter does not take effect now.
string identity_token	Information about the token used for the registry authentication. This parameter does not take effect now.
string registry_token	Information about the token used for the interaction with the registry. This parameter does not take effect now.

- **Container**

  The API is used to describe container information, such as the ID and status.

Parameter	Description
string id	Container ID.
string pod_sandbox_id	ID of the sandbox to which the container belongs.
ContainerMetadata metadata	Container metadata.
ImageSpec image	Image specifications.
string image_ref	Image used by the container. This parameter is an image ID for most runtime.
ContainerState state	Container status.
int64 created_at	Container creation timestamp (unit: ns).
map<string, string> labels	Key-value pair that can be used to identify a container or a series of containers.
map<string, string> annotations	Key-value pair that stores any information, whose values cannot be changed by the runtime.

- **ContainerStatus**

  The API is used to describe the container status information.

Parameter	Description
string id	Container ID.
ContainerMetadata metadata	Container metadata.
ContainerState state	Container status.
int64 created_at	Container creation timestamp (unit: ns).
int64 started_at	Container start timestamp (unit: ns).
int64 finished_at	Container exit timestamp (unit: ns).
int32 exit_code	Container exit code.
ImageSpec image	Image specifications.
string image_ref	Image used by the container. This parameter is an image ID for most runtime.
string reason	Brief description of the reason why the container is in the current status.
string message	Information that is easy to read and indicates the reason why the container is in the current status.
map<string, string> labels	Key-value pair that can be used to identify a container or a series of containers.
map<string, string> annotations	Key-value pair that stores any information, whose values cannot be changed by the runtime.

Parameter	Description
repeated Mount mounts	Information about the container mount point.
string log_path	Path of the container log file that is in the **log_directory** folder configured in PodSandboxConfig.

- **ContainerStatsFilter**

  The API is used to add filter criteria for the container stats list. The intersection of multiple filter criteria is displayed.

Parameter	Description
string id	Container ID.
string pod_sandbox_id	Sandbox ID.
map<string, string> label_selector	Container label, which does not support regular expressions and must be fully matched.

- **ContainerStats**

  The API is used to add filter criteria for the container stats list. The intersection of multiple filter criteria is displayed.

Parameter	Description
ContainerAttributes attributes	Container information.
CpuUsage cpu	CPU usage information.
MemoryUsage memory	Memory usage information.
FilesystemUsage writable_layer	Information about the writable layer usage.

- **ContainerAttributes**

  The API is used to list basic container information.

Parameter	Description
string id	Container ID.
ContainerMetadata metadata	Container metadata.
map<string,string> labels	Key-value pair that can be used to identify a container or a series of containers.
map<string,string> annotations	Key-value pair that stores any information, whose values cannot be changed by the runtime.

- **CpuUsage**

The API is used to list the CPU usage information of a container.

Parameter	Description
int64 timestamp	Timestamp.
UInt64Value usage_core_nano_seconds	CPU usage (unit: ns).

- **MemoryUsage**

  The API is used to list the memory usage information of a container.

Parameter	Description
int64 timestamp	Timestamp.
UInt64Value working_set_bytes	Memory usage.

- **FilesystemUsage**

  The API is used to list the read/write layer information of a container.

Parameter	Description
int64 timestamp	Timestamp.
StorageIdentifier storage_id	Writable layer directory.
UInt64Value used_bytes	Number of bytes occupied by images at the writable layer.
UInt64Value inodes_used	Number of inodes occupied by images at the writable layer.

- **Device**

  The API is used to specify the host volume to be mounted to a container.

Parameter	Description
string container_path	Mounting path of a container.
string host_path	Mounting path on the host.
string permissions	Cgroup permission of a device. (**r** indicates that containers can be read from a specified device. **w** indicates that containers can be written to a specified device. **m** indicates that containers can create new device files.)

- **LinuxContainerConfig**

  The API is used to specify Linux configurations.

Parameter	Description
LinuxContainerResources resources	Container resource specifications.
LinuxContainerSecurityContext security_context	Linux container security configuration.

- **ContainerConfig**

  The API is used to specify all mandatory and optional fields for creating a container.

Parameter	Description
ContainerMetadata metadata	Container metadata. The information will uniquely identify a container and should be used at runtime to ensure correct operations. The information can also be used at runtime to optimize the user experience (UX) design, for example, construct a readable name. This parameter is mandatory.
ImageSpec image	Image used by the container. This parameter is mandatory.
repeated string command	Command to be executed. Default value: **/bin/sh**
repeated string args	Parameters of the command to be executed.
string working_dir	Current working path of the command.
repeated KeyValue envs	Environment variables configured in the container.
repeated Mount mounts	Information about the mount point to be mounted in the container.
repeated Device devices	Information about the device to be mapped in the container.
map<string, string> labels	Key-value pair that can be used to index and select a resource.
map<string, string> annotations	Unstructured key-value mappings that can be used to store and retrieve any metadata.
string log_path	Relative path to **PodSandboxConfig.LogDirectory**, which is used to store logs (STDOUT and STDERR) on the container host.
bool stdin	Whether to open **stdin** of the container.
bool stdin_once	Whether to immediately disconnect other data flows connected with **stdin** when a data flow connected with **stdin** is disconnected. This parameter does not take effect now.
bool tty	Whether to use a pseudo terminal to connect to **stdio** of the container.
LinuxContainerConfig linux	Container configuration information in the Linux

	system.

- **NetworkConfig**

  This API is used to specify runtime network configurations.

Parameter	Description
string pod_cidr	CIDR used by pod IP addresses.

- **RuntimeConfig**

  This API is used to specify runtime network configurations.

Parameter	Description
NetworkConfig network_config	Runtime network configurations.

- **RuntimeCondition**

  The API is used to describe runtime status information.

Parameter	Description
string type	Runtime status type.
bool status	Runtime status.
string reason	Brief description of the reason for the runtime status change.
string message	Message with high readability, which indicates the reason for the runtime status change.

- **RuntimeStatus**

  The API is used to describe runtime status.

Parameter	Description
repeated RuntimeCondition conditions	List of current runtime status.

1.3.5.2.1 Runtime Service

The runtime service provides APIs for operating pods and containers, and APIs for querying the configuration and status information of the runtime service.

?.1.RunPodSandbox

Prototype

```
rpc RunPodSandbox(RunPodSandboxRequest) returns (RunPodSandboxResponse) {}
```

Description

This API is used to create and start a PodSandbox. If the PodSandbox is successfully run, the sandbox is in the ready state.

Precautions

1. The default image for starting a sandbox is **rnd-dockerhub.huawei.com/library/pause-${***machine***}:3.0** where **${***machine***}** indicates the architecture. On x86_64, the value of *machine* is **amd64**. On ARM64, the value of *machine* is **aarch64**. Currently, only the **amd64** or **aarch64** image can be downloaded from the rnd-dockerhub registry. If the image does not exist on the host, ensure that the host can download the image from the rnd-dockerhub registry. If you want to use another image, refer to **pod-sandbox-image** in the *iSulad Deployment Configuration*.
2. The container name is obtained from fields in PodSandboxMetadata and separated by underscores (_). Therefore, the metadata cannot contain underscores (_). Otherwise, the ListPodSandbox API cannot be used for query even when the sandbox is running successfully.

Parameters

Parameter	Description
PodSandboxConfig config	Sandbox configuration.
string runtime_handler	Runtime for the created sandbox. Currently, lcr and kata-runtime are supported.

Return Values

Return Value	Description
string pod_sandbox_id	If the operation is successful, the response is returned.

?.2.StopPodSandbox

Prototype

```
rpc StopPodSandbox(StopPodSandboxRequest) returns (StopPodSandboxResponse) {}
```

Description

This API is used to stop PodSandboxes and sandbox containers, and reclaim the network resources (such as IP addresses) allocated to a sandbox. If any running container belongs to the sandbox, the container must be forcibly stopped.

Parameters

Parameter	Description
string pod_sandbox_id	Sandbox ID.

Return Values

Return Value	Description
None	None

?.3.RemovePodSandbox

Prototype

```
rpc RemovePodSandbox(RemovePodSandboxRequest) returns (RemovePodSandboxResponse) {}
```

Description

This API is used to delete a sandbox. If any running container belongs to the sandbox, the container must be forcibly stopped and deleted. If the sandbox has been deleted, no errors will be returned.

Precautions

1. When a sandbox is deleted, network resources of the sandbox are not deleted. Before deleting a pod, you must call StopPodSandbox to clear network resources. Ensure that StopPodSandbox is called at least once before deleting the sandbox.

Parameters

Parameter	Description
string pod_sandbox_id	Sandbox ID.

Return Values

Return Value	Description
None	None

?.4.PodSandboxStatus

Prototype

```
rpc PodSandboxStatus(PodSandboxStatusRequest) returns (PodSandboxStatusResponse) {}
```

Description

This API is used to query the sandbox status. If the sandbox does not exist, an error is returned.

Parameters

Parameter	Description
string pod_sandbox_id	Sandbox ID
bool verbose	Whether to display additional information about the sandbox. This parameter does not take effect now.

Return Values

Return Value	Description
PodSandboxStatus status	Status of the sandbox.
map<string, string> info	Additional information about the sandbox. The key can be any string, and the value is a JSON character string. The information can be any debugging content. When **verbose** is set to **true**, **info** cannot be empty. This parameter does not take effect now.

?.5.ListPodSandbox

Prototype

```
rpc ListPodSandbox(ListPodSandboxRequest) returns (ListPodSandboxResponse) {}
```

Description

This API is used to return the sandbox information list. Filtering based on criteria is supported.

Parameters

Parameter	Description
PodSandboxFilter filter	Filter criteria.

Return Values

Return Value	Description
repeated PodSandbox items	Sandbox information list.

?.6.CreateContainer

```
grpc::Status CreateContainer(grpc::ServerContext *context, const
runtime::CreateContainerRequest *request, runtime::CreateContainerResponse *reply) {}
```

Description

This API is used to create a container in the PodSandbox.

Precautions

- **sandbox_config** in **CreateContainerRequest** is the same as the configuration transferred to **RunPodSandboxRequest** to create a PodSandbox. It is transferred again for reference only. PodSandboxConfig must remain unchanged throughout the lifecycle of a pod.
- The container name is obtained from fields in ContainerMetadata and separated by underscores (_). Therefore, the metadata cannot contain underscores (_). Otherwise, the ListContainers API cannot be used for query even when the sandbox is running successfully.
- **CreateContainerRequest** does not contain the **runtime_handler** field. The runtime type of the container is the same as that of the corresponding sandbox.

Parameters

Parameter	Description
string pod_sandbox_id	ID of the PodSandbox where a container is to be created.
ContainerConfig config	Container configuration information.
PodSandboxConfig sandbox_config	PodSandbox configuration information.

Supplement

Unstructured key-value mappings that can be used to store and retrieve any metadata. The field can be used to transfer parameters for the fields for which the CRI does not provide specific parameters.

- Customize the field:

Custom key:value	Description

cgroup.pids.max:int64_t	Used to limit the number of processes or threads in a container. (Set the parameter to **-1** for unlimited number.)

Return Values

Return Value	Description
string container_id	ID of the created container.

?.7.StartContainer

Prototype

```
rpc StartContainer(StartContainerRequest) returns (StartContainerResponse) {}
```

Description

This API is used to start a container.

Parameters

Parameter	Description
string container_id	Container ID.

Return Values

Return Value	Description
None	None

?.8.StopContainer

Prototype

```
rpc StopContainer(StopContainerRequest) returns (StopContainerResponse) {}
```

Description

This API is used to stop a running container. You can set a graceful timeout time. If the container has been stopped, no errors will be returned.

Parameters

Parameter	Description
string container_id	Container ID.
int64 timeout	Waiting time before a container is forcibly stopped. The default value is **0**, indicating forcible stop.

Return Values

None

?.9.RemoveContainer

Prototype

```
rpc RemoveContainer(RemoveContainerRequest) returns (RemoveContainerResponse) {}
```

Description

This API is used to delete a container. If the container is running, it must be forcibly stopped. If the container has been deleted, no errors will be returned.

Parameters

Parameter	Description
string container_id	Container ID.

Return Values

None

?.10.ListContainers

Prototype

```
rpc ListContainers(ListContainersRequest) returns (ListContainersResponse) {}
```

Description

This API is used to return the container information list. Filtering based on criteria is supported.

Parameters

Parameter	Description
ContainerFilter filter	Filter criteria.

Return Values

Return Value	Description
repeated Container containers	Container information list.

?.11.ContainerStatus

Prototype

```
rpc ContainerStatus(ContainerStatusRequest) returns (ContainerStatusResponse) {}
```

Description

This API is used to return the container status information. If the container does not exist, an error will be returned.

Parameters

Parameter	Description
string container_id	Container ID.
bool verbose	Whether to display additional information about the sandbox. This parameter does not take effect now.

Return Values

Return Value	Description
ContainerStatus status	Container status information.
map<string, string> info	Additional information about the sandbox. The key can be any string, and the value is a JSON character string. The information can be any debugging content. When **verbose** is set to **true**, **info** cannot be empty. This parameter does not take effect now.

?.12.UpdateContainerResources

Prototype

```
rpc UpdateContainerResources(UpdateContainerResourcesRequest) returns
(UpdateContainerResourcesResponse) {}
```

Description

This API is used to update container resource configurations.

Precautions

- This API cannot be used to update the pod resource configurations.
- The value of **oom_score_adj** of any container cannot be updated.

Parameters

Parameter	Description
string container_id	Container ID.
LinuxContainerResources linux	Linux resource configuration information.

Return Values

None

?.13.ExecSync

Prototype

```
rpc ExecSync(ExecSyncRequest) returns (ExecSyncResponse) {}
```

Description

The API is used to run a command in containers in synchronization mode through the gRPC communication method.

Precautions

The interaction between the terminal and the containers must be disabled when a single command is executed.

Parameters

Parameter	Description
string container_id	Container ID.
repeated string cmd	Command to be executed.
int64 timeout	Timeout period for stopping the command (unit: second). The default value is **0**, indicating that there is no timeout limit. This parameter does not take effect now.

Return Values

Return Value	Description
bytes stdout	Standard output of the capture command.
bytes stderr	Standard error output of the capture command.
int32 exit_code	Exit code, which represents the completion of command execution. The default value is **0**, indicating that the command is executed successfully.

?.14.Exec

Prototype

```
rpc Exec(ExecRequest) returns (ExecResponse) {}
```

Description

This API is used to run commands in a container through the gRPC communication method, that is, obtain URLs from the CRI server, and then use the obtained URLs to establish a long connection to the WebSocket server, implementing the interaction with the container.

Precautions

The interaction between the terminal and the container can be enabled when a single command is executed. One of **stdin**, **stdout**, and **stderr** must be true. If **tty** is true, **stderr** must be false. Multiplexing is not supported. In this case, the output of **stdout** and **stderr** will be combined to a stream.

Parameters

Parameter	Description
string container_id	Container ID.
repeated string cmd	Command to be executed.
bool tty	Whether to run the command in a TTY.
bool stdin	Whether to generate the standard input stream.
bool stdout	Whether to generate the standard output stream.
bool stderr	Whether to generate the standard error output stream.

Return Values

Return Value	Description
string url	Fully qualified URL of the exec streaming server.

?.15.Attach

Prototype

```
rpc Attach(AttachRequest) returns (AttachResponse) {}
```

Description

This API is used to take over the init process of a container through the gRPC communication method, that is, obtain URLs from the CRI server, and then use the obtained URLs to establish a long connection to the WebSocket server, implementing the interaction with the container. Only containers whose runtime is of the LCR type are supported.

Parameters

Parameter	Description
string container_id	Container ID.
bool tty	Whether to run the command in a TTY.
bool stdin	Whether to generate the standard input stream.
bool stdout	Whether to generate the standard output stream.
bool stderr	Whether to generate the standard error output stream.

Return Values

Return Value	Description
string url	Fully qualified URL of the attach streaming server.

?.16.ContainerStats

Prototype

```
rpc ContainerStats(ContainerStatsRequest) returns (ContainerStatsResponse) {}
```

Description

This API is used to return information about resources occupied by a container. Only containers whose runtime is of the LCR type are supported.

Parameters

Parameter	Description
string container_id	Container ID.

Return Values

Return Value	Description
ContainerStats stats	Container information. Note: Disks and inodes support only the query of containers started by OCI images.

?.17.ListContainerStats

Prototype

```
rpc ListContainerStats(ListContainerStatsRequest) returns
(ListContainerStatsResponse) {}
```

Description

This API is used to return the information about resources occupied by multiple containers. Filtering based on criteria is supported.

Parameters

Parameter	Description
ContainerStatsFilter filter	Filter criteria.

Return Values

Return Value	Description
repeated ContainerStats stats	Container information list. Note: Disks and inodes support only the query of containers started by OCI images.

?.18.UpdateRuntimeConfig

Prototype

```
rpc UpdateRuntimeConfig(UpdateRuntimeConfigRequest) returns
(UpdateRuntimeConfigResponse);
```

Description

This API is used as a standard CRI to update the pod CIDR of the network plug-in. Currently, the CNI network plug-in does not need to update the pod CIDR. Therefore, this API records only access logs.

Precautions

API operations will not modify the system management information, but only record a log.

Parameters

Parameter	Description
RuntimeConfig runtime_config	Information to be configured for the runtime.

Return Values

None

?.19.Status

Prototype

```
rpc Status(StatusRequest) returns (StatusResponse) {};
```

Description

This API is used to obtain the network status of the runtime and pod. Obtaining the network status will trigger the update of network configuration. Only containers whose runtime is of the LCR type are supported.

Precautions

If the network configuration fails to be updated, the original configuration is not affected. The original configuration is overwritten only when the update is successful.

Parameters

Parameter	Description
bool verbose	Whether to display additional runtime information. This parameter does not take effect now.

Return Values

Return Value	Description
RuntimeStatus status	Runtime status.
map<string, string> info	Additional information about the runtime. The key of **info** can be any value. The value must be in JSON format and can contain any debugging information. When **verbose** is set to **true**, **info** cannot be empty.

1.3.5.2.2 Image Service

The service provides the gRPC API for pulling, viewing, and removing images from the registry.

?.1.ListImages

Prototype

```
rpc ListImages(ListImagesRequest) returns (ListImagesResponse) {}
```

Description

This API is used to list existing image information.

Precautions

This is a unified API. You can run the **cri images** command to query embedded images. However, embedded images are not standard OCI images. Therefore, query results have the following restrictions:

- An embedded image does not have an image ID. Therefore, the value of **image ID** is the config digest of the image.
- An embedded image has only config digest, and it does not comply with the OCI image specifications. Therefore, the value of **digest** cannot be displayed.

Parameters

Parameter	Description
ImageSpec filter	Name of the image to be filtered.

Return Values

Return Value	Description
repeated Image images	Image information list.

?.2.ImageStatus

Prototype

```
rpc ImageStatus(ImageStatusRequest) returns (ImageStatusResponse) {}
```

Description

The API is used to query the information about a specified image.

Precautions

1. If the image to be queried does not exist, **ImageStatusResponse** is returned and **Image** is set to **nil** in the return value.
2. This is a unified API. Since embedded images do not comply with the OCI image specifications and do not contain required fields, the images cannot be queried by using this API.

Parameters

Parameter	Description
ImageSpec image	Image name.
bool verbose	Whether to query additional information. This parameter does not take effect now. No additional information is returned.

Return Values

Return Value	Description
Image image	Image information.
map<string, string> info	Additional image information. This parameter does not take effect now. No additional information is returned.

?.3.PullImage

Prototype

```
rpc PullImage(PullImageRequest) returns (PullImageResponse) {}
```

Description

This API is used to download images.

Precautions

Currently, you can download public images, and use the username, password, and auth information to download private images. The **server_address**, **identity_token**, and **registry_token** fields in **authconfig** cannot be configured.

Parameters

Parameter	Description
ImageSpec image	Name of the image to be downloaded.
AuthConfig auth	Verification information for downloading a private

	image.
PodSandboxConfig sandbox_config	Whether to download an image in the pod context. This parameter does not take effect now.

Return Values

Return Value	Description
string image_ref	Information about the downloaded image.

?.4.RemoveImage

Prototype

```
rpc RemoveImage(RemoveImageRequest) returns (RemoveImageResponse) {}
```

Description

This API is used to delete specified images.

Precautions

This is a unified API. Since embedded images do not comply with the OCI image specifications and do not contain required fields, you cannot delete embedded images by using this API and the image ID.

Parameters

Parameter	Description
ImageSpec image	Name or ID of the image to be deleted.

Return Values

None

?.5.ImageFsInfo

Prototype

```
rpc ImageFsInfo(ImageFsInfoRequest) returns (ImageFsInfoResponse) {}
```

Description

This API is used to query the information about the file system that stores images.

Precautions

Queried results are the file system information in the image metadata.

Parameters

None

Return Values

Return Value	Description
repeated FilesystemUsage image_filesystems	Information about the file system that stores images.

1.3.5.3 Constraints

1. If **log_directory** is configured in the **PodSandboxConfig** parameter when a sandbox is created, **log_path** must be specified in **ContainerConfig** when all containers that belong to the sandbox are created. Otherwise, the containers may not be started or deleted by using the CRI.

   The actual value of **LOGPATH** of containers is **log_directory/log_path**. If **log_path** is not set, the final value of **LOGPATH** is changed to **log_directory**.

   - If the path does not exist, iSulad will create a soft link pointing to the actual path of container logs when starting a container. Then **log_directory** becomes a soft link. There are two cases:

     i. In the first case, if **log_path** is not configured for other containers in the sandbox, **log_directory** will be deleted and point to **log_path** of the newly started container. As a result, logs of the first started container point to logs of the later started container.

     ii. In the second case, if **log_path** is configured for other containers in the sandbox, the value of **LOGPATH** of the container is **log_directory/log_path**. Because **log_directory** is a soft link, the creation fails when **log_directory/log_path** is used as the soft link to point to the actual path of container logs.

   - If the path exists, iSulad will attempt to delete the path (non-recursive) when starting a container. If the path is a folder path containing content, the deletion fails. As a result, the soft link fails to be created, the container fails to be started, and the same error occurs when the container is going to be deleted.

2. If **log_directory** is configured in the **PodSandboxConfig** parameter when a sandbox is created, and **log_path** is specified in **ContainerConfig** when a container is created, the final value of **LOGPATH** is **log_directory/log_path**. iSulad does not recursively create **LOGPATH**, therefore, you must ensure that **dirname(LOGPATH)** exists, that is, the upper-level path of the final log file path exists.

3. If **log_directory** is configured in the **PodSandboxConfig** parameter when a sandbox is created, and the same **log_path** is specified in **ContainerConfig** when multiple containers are created, or if containers in different sandboxes point to the same **LOGPATH**, the latest container log path will overwrite the previous path after the containers are started successfully.

4. If the image content in the remote registry changes and the original image is stored in the local host, the name and tag of the original image are changed to **none** when you call the CRI Pull image API to download the image again.

   An example is as follows:

   Locally stored images:

   ```
   IMAGE                                  TAG              IMAGE ID          SIZE
   rnd-dockerhub.huawei.com/pproxyisulad/test   latest           99e59f495ffaa
   753kB
   ```

   After the **rnd-dockerhub.huawei.com/pproxyisulad/test:latest** image in the remote registry is updated and downloaded again:

   ```
   IMAGE                                  TAG              IMAGE ID          SIZE
   <none>                                 <none>           99e59f495ffaa
   753kB
   rnd-dockerhub.huawei.com/pproxyisulad/test   latest           d8233ab899d41
   1.42MB
   ```

   Run the **isula images** command. The value of **REF** is displayed as **-**.

   ```
   REF                                        IMAGE ID            CREATED
   SIZE
   rnd-dockerhub.huawei.com/pproxyisulad/test:latest d8233ab899d41
   2019-02-14 19:19:37  1.42MB
   -                                          99e59f495ffaa        2016-05-04
   02:26:41  753kB
   ```

1.3.6 Image Management

1.3.6.1 Docker Image Management

1.3.6.1.1 Logging In to a Registry

Description

The **isula login** command is run to log in to a registry. After successful login, you can run the **isula pull** command to pull images from the registry. If the registry does not require a password, you do not need to run this command before pulling images.

Usage

```
isula login [OPTIONS] SERVER
```

Parameters

For details about parameters in the **login** command, see Table 1-21.

Example

```
$ isula login -u abc my.csp-edge.com:5000

Login Succeeded
```

1.3.6.1.2 Logging Out of a Registry

Description

The **isula logout** command is run to log out of a registry. If you run the **isula pull** command to pull images from the registry after logging out of the system, the image will fail to be pulled because you are not authenticated.

Usage

```
isula logout SERVER
```

Parameters

For details about parameters in the **logout** command, see Table 1-22.

Example

```
$ isula logout my.csp-edge.com:5000
Logout Succeeded
```

1.3.6.1.3 Pulling Images from a Registry

Description

Pull images from a registry to the local host.

Usage

```
isula pull [OPTIONS] NAME[:TAG|@DIGEST]
```

Parameters

For details about parameters in the **pull** command, see Table 1-23.

Example

```
$ isula pull localhost:5000/official/busybox
Image "localhost:5000/official/busybox" pulling
Image
"localhost:5000/official/busybox@sha256:bf510723d2cd2d4e3f5ce7e93bf1e52c8fd76831995ac3bd3f90ecc866643aff" pulled
```

1.3.6.1.4 Deleting Images

Description

Delete one or more images.

Usage

```
isula rmi [OPTIONS] IMAGE [IMAGE...]
```

Parameters

For details about parameters in the **rmi** command, see Table 1-24.

Example

```
$ isula rmi rnd-dockerhub.huawei.com/official/busybox
Image "rnd-dockerhub.huawei.com/official/busybox" removed
```

1.3.6.1.5 Loading Images

Description

Load images from a .tar package. The .tar package must be exported by using the **docker save** command or must be in the same format.

Usage

```
isula load [OPTIONS]
```

Parameters

For details about parameters in the **load** command, see Table 1-25.

Example

```
$ isula load -i busybox.tar
Load image from "/root/busybox.tar" success
```

1.3.6.1.6 Listing Images

Description

List all images in the current environment.

Usage

```
isula images
```

Parameters

For details about parameters in the **images** command, see Table 1-26.

Example

```
$ isula images
REF                                          IMAGE ID           CREATED            SIZE
rnd-dockerhub.huawei.com/official/busybox:latest e4db68de4ff2       2019-06-15
08:19:54  1.376 MB
```

1.3.6.1.7 Inspecting Images

Description

After the configuration information of an image is returned, you can use the **-f** parameter to filter the information as needed.

Usage

```
isula inspect [options] CONTAINER|IMAGE [CONTAINER|IMAGE...]
```

Parameters

For details about parameters in the **inspect** command, see Table 1-27.

Example

```
$ isula inspect -f "{{json .image.id}}" rnd-dockerhub.huawei.com/official/busybox
"e4db68de4ff27c2adfea0c54bbb73a61a42f5b667c326de4d7d5b19ab71c6a3b"
```

1.3.6.1.8 Two-Way Authentication

Description

After this function is enabled, iSulad and image repositories communicate over HTTPS. Both iSulad and image repositories verify the validity of each other.

Usage

The corresponding registry needs to support this function and iSulad needs to be configured as follows:

1. Modify iSulad configuration (default path: **/etc/isulad/daemon.json**) and set **use-decrypted-key** to **false**.
2. Place related certificates in the folder named after the registry in the **/etc/isulad/certs.d** directory. For details about how to generate certificates, visit the official Docker website:
   - https://docs.docker.com/engine/security/certificates/
   - https://docs.docker.com/engine/security/https/
3. Run the **systemctl restart isulad** command to restart iSulad.

Parameters

Parameters can be configured in the **/etc/isulad/daemon.json** file or carried when iSulad is started.

```
isulad --use-decrypted-key=false
```

Example

Set **use-decrypted-key** to **false**.

```
$ cat /etc/isulad/daemon.json
{
   "group": "isulad",
```

```
   "graph": "/var/lib/isulad",
   "state": "/var/run/isulad",
   "engine": "lcr",
   "log-level": "ERROR",
   "pidfile": "/var/run/isulad.pid",
   "log-opts": {
      "log-file-mode": "0600",
      "log-path": "/var/lib/isulad",
      "max-file": "1",
      "max-size": "30KB"
   },
   "log-driver": "stdout",
   "hook-spec": "/etc/default/isulad/hooks/default.json",
   "start-timeout": "2m",
   "storage-driver": "overlay2",
   "storage-opts": [
      "overlay2.override kernel check=true"
   ],
   "registry-mirrors": [
      "docker.io"
   ],
   "insecure-registries": [
      "rnd-dockerhub.huawei.com"
   ],
   "pod-sandbox-image": "",
   "image-opt-timeout": "5m",
   "native.umask": "secure",
   "network-plugin": "",
   "cni-bin-dir": "",
   "cni-conf-dir": "",
   "image-layer-check": false,
   "use-decrypted-key": false,
   "insecure-skip-verify-enforce": false
}
```

Place the certificate in the corresponding directory.

```
$ pwd
/etc/isulad/certs.d/my.csp-edge.com:5000
$ ls
ca.crt  tls.cert  tls.key
```

Restart iSulad.

```
$ systemctl restart isulad
```

Run the **pull** command to download images from the registry:

```
$ isula pull my.csp-edge.com:5000/busybox
Image "my.csp-edge.com:5000/busybox" pulling
Image
"my.csp-edge.com:5000/busybox@sha256:f1bdc62115dbfe8f54e52e19795ee34b4473babdeb9bc
4f83045d85c7b2ad5c0" pulled
```

1.3.6.2 Embedded Image Management

1.3.6.2.1 Loading Images

Description

Load images based on the **manifest** files of embedded images. The value of **--type** must be set to **embedded**.

Usage

```
isula load [OPTIONS] --input=FILE --type=TYPE
```

Parameters

For details about parameters in the **load** command, see Table 1-25.

Example

```
$ isula load -i test.manifest --type embedded
Load image from "/root/work/bugfix/tmp/ci testcase data/embedded/img/test.manifest"
success
```

1.3.6.2.2 Listing Images

Description

List all images in the current environment.

Usage

```
isula images [OPTIONS]
```

Parameters

For details about parameters in the **images** command, see Table 1-26.

Example

```
$ isula images
REF                          IMAGE ID           CREATED            SIZE
test:v1                       9319da1f5233       2018-03-01 10:55:44  1.273 MB
```

1.3.6.2.3 Inspecting Images

Description

After the configuration information of an image is returned, you can use the **-f** parameter to filter the information as needed.

Usage

```
isula inspect [options] CONTAINER|IMAGE [CONTAINER|IMAGE...]
```

Parameters

For details about parameters in the **inspect** command, see Table 1-27.

Example

```
$ isula inspect -f "{{json .created}}" test:v1
"2018-03-01T15:55:44.322987811Z"
```

1.3.6.2.4 Deleting Images

Description

Delete one or more images.

Usage

```
isula rmi [OPTIONS] IMAGE [IMAGE...]
```

Parameters

For details about parameters in the **rmi** command, see Table 1-24.

Example

```
$ isula rmi test:v1
Image "test:v1" removed
```

1.3.7 Checking the Container Health Status

1.3.7.1 Scenarios

In the production environment, bugs are inevitable in applications provided by developers or services provided by platforms. Therefore, a management system is indispensable for periodically checking and repairing applications. The container health check mechanism adds a user-defined health check function for containers. When a container is created, the **--health-cmd** option is configured so that commands are periodically executed in the container to monitor the health status of the container based on return values.

1.3.7.2 Configuration Methods

Configurations during container startup:

```
isula run -itd --health-cmd "echo iSulad >> /tmp/health check file || exit 1"
--health-interval 5m --health-timeout 3s --health-exit-on-unhealthy  busybox bash
```

The configurable options are as follows:

- **--health-cmd**: This option is mandatory. If **0** is returned after a command is run in a container, the command execution succeeds. If a value other than **0** is returned, the command execution fails.
- **--health-interval**: interval between two consecutive command executions. The default value is **30s**. The value ranges from **1s** to the maximum value of Int64 (unit: nanosecond). If the input parameter is set to **0s**, the default value is used.

- **--health-timeout**: maximum duration for executing a single check command. If the execution times out, the command execution fails. The default value is **30s**. The value ranges from **1s** to the maximum value of Int64 (unit: nanosecond). If the input parameter is set to **0s**, the default value is used. Only containers whose runtime is of the LCR type are supported.
- **--health-start-period**: container initialization time. The default value is **0s**. The value ranges from **1s** to the maximum value of Int64 (unit: nanosecond).
- **--health-retries**: maximum number of retries for the health check. The default value is **3**. The maximum value is the maximum value of Int32.
- **--health-exit-on-unhealthy**: specifies whether to kill a container when it is unhealthy. The default value is **false**.

1.3.7.3 Check Rules

1. After a container is started, the container status is **health:starting**.
2. After the period specified by **start-period**, the **cmd** command is periodically executed in the container at the interval specified by **interval**. That is, after the command is executed, the command will be executed again after the specified period.
3. If the **cmd** command is successfully executed within the time specified by **timeout** and the return value is **0**, the check is successful. Otherwise, the check fails. If the check is successful, the container status changes to **health:healthy**.
4. If the **cmd** command fails to be executed for the number of times specified by **retries**, the container status changes to **health:unhealthy**, and the container continues the health check.
5. When the container status is **health:unhealthy**, the container status changes to **health:healthy** if a check succeeds.
6. If **--exit-on-unhealthy** is set, and the container exits due to reasons other than being killed (the returned exit code is **137**), the health check takes effect only after the container is restarted.
7. When the **cmd** command execution is complete or times out, Docker daemon will record the start time, return value, and standard output of the check to the configuration file of the container. A maximum of five records can be recorded. In addition, the configuration file of the container stores health check parameters.
8. When the container is running, the health check status is written into the container configurations. You can run the **isula inspect** command to view the status.

```
"Health": {
   "Status": "healthy",
   "FailingStreak": 0,
   "Log": [
      {
         "Start": "2018-03-07T07:44:15.481414707-05:00",
         "End": "2018-03-07T07:44:15.556908311-05:00",
         "ExitCode": 0,
         "Output": ""
      },
      {
         "Start": "2018-03-07T07:44:18.557297462-05:00",
         "End": "2018-03-07T07:44:18.63035891-05:00",
         "ExitCode": 0,
         "Output": ""
      },
```

```
        ......
}
```

1.3.7.4 Usage Restrictions

- A maximum of five health check status records can be stored in a container. The last five records are saved.
- If health check parameters are set to **0** during container startup, the default values are used.
- After a container with configured health check parameters is started, if iSulad daemon exits, the health check is not executed. After iSulad daemon is restarted, the health status of the running container changes to **starting**. Afterwards, the check rules are the same as above.
- If the health check fails for the first time, the health check status will not change from **starting** to **unhealthy** until the specified number of retries (**--health-retries**) is reached, or to **healthy** until the health check succeeds.
- The health check function of containers whose runtime is of the Open Container Initiative (OCI) type needs to be improved. Only containers whose runtime is of the LCR type are supported.

1.3.8 Querying Information

1.3.8.1 Querying the Service Version

Description

The **isula version** command is run to query the version of the iSulad service.

Usage

```
isula version
```

Example

Query the version information.

```
isula version
```

If the iSulad service is running properly, you can view the information about versions of the client, server, and **OCI config**.

```
Client:
  Version:     1.0.31
  Git commit:   fa7f9902738e8b3d7f2eb22768b9a1372ddd1199
  Built:       2019-07-30T04:21:48.521198248-04:00

Server:
  Version:     1.0.31
  Git commit:   fa7f9902738e8b3d7f2eb22768b9a1372ddd1199
  Built:       2019-07-30T04:21:48.521198248-04:00

OCI config:
  Version:     1.0.0-rc5-dev
  Default file: /etc/default/isulad/config.json
```

If the iSulad service is not running, only the client information is queried and a message is displayed indicating that the connection times out.

```
Client:
  Version:     1.0.31
  Git commit:   fa7f9902738e8b3d7f2eb22768b9a1372ddd1199
  Built:       2019-07-30T04:21:48.521198248-04:00

Can not connect with server.Is the iSulad daemon running on the host?
```

Therefore, the **isula version** command is often used to check whether the iSulad service is running properly.

1.3.8.2 Querying System-level Information

Description

The **isula info** command is run to query the system-level information, number of containers, and number of images.

Usage

```
isula info
```

Example

Query system-level information, including the number of containers, number of images, kernel version, and operating system (OS).

```
$ isula info
Containers: 2
 Running: 0
 Paused: 0
 Stopped: 2
Images: 8
Server Version: 1.0.31
Logging Driver: json-file
Cgroup Driverr: cgroupfs
Hugetlb Pagesize: 2MB
Kernel Version: 4.19
Operating System: Fedora 29 (Twenty Nine)
OSType: Linux
Architecture: x86 64
CPUs: 8
Total Memory: 7 GB
Name: localhost.localdomain
iSulad Root Dir: /var/lib/isulad
```

1.3.9 Security Features

1.3.9.1 Seccomp Security Configuration

1.3.9.1.1 Scenarios

Secure computing mode (seccomp) is a simple sandboxing mechanism introduced to the Linux kernel from version 2.6.23. In some specific scenarios, you may want to perform some privileged operations in a container without starting the privileged container. You can add **--cap-add** at runtime to obtain some small-scope permissions. For container instances with strict security requirements, th capability granularity may not meet the requirements. You can use some methods to control the permission scope in a refined manner.

- Example

  In a common container scenario, you can use the **-v** flag to map a directory (including a binary file that cannot be executed by common users) on the host to the container.

  In the container, you can add chmod 4777 (the modification permission of the binary file) to the S flag bit. In this way, on the host, common users who cannot run the binary file (or whose running permission is restricted) can obtain the permissions of the binary file (such as the root permission) when running the binary file after the action added to the S flag bit is performed, so as to escalate the permission or access other files.

  In this scenario, if strict security requirements are required, the chmod, fchmod, and fchmodat system calls need to be tailored by using seccomp.

1.3.9.1.2 Usage Restrictions

- Seccomp may affect performance. Before setting seccomp, evaluate the scenario and add the configuration only if necessary.

1.3.9.1.3 Usage Guide

Use **--security-opt** to transfer the configuration file to the container where system calls need to be filtered.

```
isula run -itd --security-opt seccomp=/path/to/seccomp/profile.json
rnd-dockerhub.huawei.com/official/busybox
```

NOTE

1. When the configuration file is transferred to the container by using **--security-opt** during container creation, the default configuration file (**/etc/isulad/seccomp_default.json**) is used.
2. When **--security-opt** is set to **unconfined** during container creation, system calls are not filtered for the container.
3. **/path/to/seccomp/profile.json** must be an absolute path.

Obtaining the Default Seccomp Configuration of a Common Container

- Start a common container (or a container with **--cap-add**) and check its default permission configuration.

  ```
  cat /etc/isulad/seccomp_default.json | python -m json.tool > profile.json
  ```

  The **seccomp** field contains many **syscalls** fields. Then extract only the **syscalls** fields and perform the customization by referring to the customization of the seccomp configuration file.

  ```
  "defaultAction": "SCMP_ACT_ERRNO",
  "syscalls": [
  ```

```
{
"action": "SCMP ACT ALLOW",
"name": "accept"
},
{
"action": "SCMP ACT ALLOW",
"name": "accept4"
},
{
"action": "SCMP ACT ALLOW",
"name": "access"
},
{
"action": "SCMP ACT ALLOW",
"name": "alarm"
},
{
"action": "SCMP ACT ALLOW",
"name": "bind"
},
]...
```

- Check the seccomp configuration that can be identified by the LXC.

```
cat
/var/lib/isulad/engines/lcr/74353e38021c29314188e29ba8c1830a4677ffe5c4decda77a1
e0853ec8197cd/seccomp
...
waitpid allow
write allow
writev allow
ptrace allow
personality allow [0,0,SCMP CMP EQ,0]
personality allow [0,8,SCMP CMP EQ,0]
personality allow [0,131072,SCMP CMP EQ,0]
personality allow [0,131080,SCMP CMP EQ,0]
personality allow [0,4294967295,SCMP CMP EQ,0]
...
```

Customizing the Seccomp Configuration File

When starting a container, use **--security-opt** to introduce the seccomp configuration file. Container instances will restrict the running of system APIs based on the configuration file. Obtain the default seccomp configuration of common containers, obtain the complete template, and customize the configuration file by referring to this section to start the container.

```
isula run --rm -it --security-opt seccomp:/path/to/seccomp/profile.json
rnd-dockerhub.huawei.com/official/busybox
```

The configuration file template is as follows:

```
{
"defaultAction": "SCMP ACT ALLOW",
"syscalls": [
{
"name": "syscall-name",
"action": "SCMP_ACT_ERRNO",
```

```
"args": null
}
]
}
```

NOTICE

- **defaultAction** and **syscalls**: The types of their corresponding actions are the same, but their values must be different. The purpose is to ensure that each syscall has a default action. Clear definitions in the syscall array shall prevail. As long as the values of **defaultAction** and **action** are different, no action conflicts will occur. The following actions are supported:

  SCMP_ACT_ERRNO: forbids calling syscalls and displays error information.

  SCMP_ACT_ALLOW: allows calling syscalls.

- **syscalls**: array, which can contain one or more syscalls. **args** is optional.
- **name**: syscalls to be filtered.
- **args**: array. The definition of each object in the array is as follows:

```
   type Arg struct {
Index    uint     `json:"index"`     // Parameter ID. Take open(fd, buf, len) as an example. The fd corresponds to 0 and buf corresponds to 1.
Value    uint64   `json:"value"`     // Value to be compared with the parameter.
ValueTwo uint64   `json:"value_two"` // It is valid only when Op is set to MaskEqualTo. After the bitwise AND operation is performed on the user-defined value and the value of Value, the result is compared with the value of ValueTwo. If they are the same, the action is executed.
Op       Operator `json:"op"`
}
```

  The value of **Op** in **args** can be any of the following:

  "SCMP_CMP_NE": NotEqualTo

  "SCMP_CMP_LT": LessThan

  "SCMP_CMP_LE": LessThanOrEqualTo

  "SCMP_CMP_EQ": EqualTo

  "SCMP_CMP_GE": GreaterThanOrEqualTo

  "SCMP_CMP_GT": GreaterThan

  "SCMP_CMP_MASKED_EQ": MaskEqualTo

1.3.9.2 capabilities Security Configuration

1.3.9.2.1 Scenarios

The capability mechanism is a security feature introduced to Linux kernel after version 2.2. The super administrator permission is controlled at a smaller granularity to prevent the root permission from being used. The root permission is divided based on different domains so that the divided permissions can be enabled or disabled separately. For details about capabilities, see the *Linux Programmer's Manual* (capabilities(7) - Linux man page).

```
man capabilities
```

1.3.9.2.2 Usage Restrictions

- The default capability list (whitelist) of the iSulad service, which is carried by common container processes by default, are as follows:

```
"CAP CHOWN",
"CAP DAC OVERRIDE",
"CAP FSETID",
"CAP FOWNER",
"CAP MKNOD",
"CAP NET RAW",
"CAP SETGID",
"CAP SETUID",
"CAP SETFCAP",
"CAP SETPCAP",
"CAP NET BIND SERVICE",
"CAP SYS CHROOT",
"CAP KILL",
"CAP_AUDIT_WRITE"
```

- Default configurations of capabilities include **CAP_SETUID** and **CAP_FSETID**. If the host and a container share a directory, the container can set permissions for the binary file in the shared directory. Common users on the host can use this feature to elevate privileges. The container can write **CAP_AUDIT_WRITE** to the host, which may cause risks. If the application scenario does not require this capability, you are advised to use **--cap-drop** to delete the capability when starting the container.
- Adding capabilities means that the container process has greater capabilities than before. In addition, more system call APIs are opened.

1.3.9.2.3 Usage Guide

iSulad uses **--cap-add** or **--cap-drop** to add or delete specific permissions for a container. Do not add extra permissions to the container unless necessary. You are advised to remove the default but unnecessary permissions from the container.

```
isula run --rm -it --cap-add all --cap-drop SYS ADMIN
rnd-dockerhub.huawei.com/official/busybox
```

1.3.9.3 SELinux Security Configuration

1.3.9.3.1 Scenarios

Security-Enhanced Linux (SELinux) is a Linux kernel security module that provides a mechanism for supporting access control security policies. Through Multi-Category Security (MCS), iSulad labels processes in containers to control containers' access to resources, reducing privilege escalation risks and preventing further damage.

1.3.9.3.2 Usage Restrictions

- Ensure that SELinux is enabled for the host and daemon (the **selinux-enabled** field in the **/etc/isulad/daemon.json** file is set to **true** or **--selinux-enabled** is added to command line parameters).
- Ensure that a proper SELinux policy has been configured on the host. container-selinux is recommended.
- The introduction of SELinux affects the performance. Therefore, evaluate the scenario before setting SELinux. Enable the SELinux function for the daemon and set the SELinux configuration in the container only when necessary.

- When you configure labels for a mounted volume, the source directory cannot be a subdirectory of **/**, **/usr**, **/etc**, **/tmp**, **/home**, **/run**, **/var**, **/root**, or **/usr**.

NOTE

- iSulad does not support labeling the container file system. To ensure that the container file system and configuration directory are labeled with the container access permission, run the **chcon** command to label them.
- If SELinux access control is enabled for iSulad, you are advised to add a label to the **/var/lib/isulad** directory before starting daemon. Files and folders generated in the directory during container creation inherit the label by default. For example:

```
chcon -R system_u:object_r:container_file_t:s0 /var/lib/isulad
```

1.3.9.3.3 Usage Guide

- Enable SELinux for daemon.

```
isulad --selinux-enabled
```

- Configure SELinux security context labels during container startup.

  --security-opt="label=user:USER": Set the label user for the container.

  --security-opt="label=role:ROLE": Set the label role for the container.

  --security-opt="label=type:TYPE": Set the label type for the container.

  --security-opt="label=level:LEVEL": Set the label level for the container.

  --security-opt="label=disable": Disable the SELinux configuration for the container.

```
$ isula run -itd --security-opt label=type:container t --security-opt
label=level:s0:c1,c2 rnd-dockerhub.huawei.com/official/centos
9be82878a67e36c826b67f5c7261c881ff926a352f92998b654bc8e1c6eec370
```

- Add the selinux label to a mounted volume (**z** indicates the shared mode).

```
$ isula run -itd -v /test:/test:z rnd-dockerhub.huawei.com/official/centos
9be82878a67e36c826b67f5c7261c881ff926a352f92998b654bc8e1c6eec370

$ls -Z /test
system_u:object_r:container_file_t:s0 file
```

1.3.10 Supporting OCI hooks

1.3.10.1 Description

The running of standard OCI hooks within the lifecycle of a container is supported. There are three types of standard hooks:

- prestart hook: executed after the **isula start** command is executed and before the init process of the container is started.
- poststart hook: executed after the init process is started and before the **isula start** command is returned.
- poststop hook: executed after the container is stopped and before the stop command is returned.

The configuration format specifications of OCI hooks are as follows:

- **path**: (Mandatory) The value must be a character string and must be an absolute path. The specified file must have the execute permission.
- **args**: (Optional) The value must be a character string array. The syntax is the same as that of **args** in **execv**.
- **env**: (Optional) The value must be a character string array. The syntax is the same as that of environment variables. The content is a key-value pair, for example, **PATH=/usr/bin**.
- **timeout**: (Optional) The value must be an integer that is greater than 0. It indicates the timeout interval for hook execution. If the running time of the hook process exceeds the configured time, the hook process is killed.

The hook configuration is in JSON format and usually stored in a file ended with **json**. An example is as follows:

```
{
       "prestart": [
           {
               "path": "/usr/bin/echo",
               "args": ["arg1", "arg2"],
               "env":  [ "key1=value1"],
               "timeout": 30
           },
           {
               "path": "/usr/bin/ls",
               "args": ["/tmp"]
           }
       ],
       "poststart": [
           {
               "path": "/usr/bin/ls",
               "args": ["/tmp"],
               "timeout": 5
           }
       ],
       "poststop": [
           {
               "path": "/tmp/cleanup.sh",
               "args": ["cleanup.sh", "-f"]
           }
       ]
}
```

1.3.10.2 APIs

Both iSulad and iSula provide the hook APIs. The default hook configurations provided by iSulad apply to all containers. The hook APIs provided by iSula apply only to the currently created container.

The default OCI hook configurations provided by iSulad are as follows:

- Set the configuration item **hook-spec** in the **/etc/isulad/daemon.json** configuration file to specify the path of the hook configuration file. Example: **"hook-spec": "/etc/default/isulad/hooks/default.json"**
- Use the **isulad --hook-spec** parameter to set the path of the hook configuration file.

The OCI hook configurations provided by iSula are as follows:

- **isula create --hook-spec**: specifies the path of the hook configuration file in JSON format.
- **isula run --hook-spec**: specifies the path of the hook configuration file in JSON format.

The configuration for **run** takes effect in the creation phase.

1.3.10.3 Usage Restrictions

- The path specified by **hook-spec** must be an absolute path.
- The file specified by **hook-spec** must exist.
- The path specified by **hook-spec** must contain a common text file in JSON format.
- The file specified by **hook-spec** cannot exceed 10 MB.
- **path** configured for hooks must be an absolute path.
- The file that is designated by **path** configured for hooks must exist.
- The file that is designated by **path** configured for hooks must have the execute permission.
- The owner of the file that is designated by **path** configured for hooks must be user **root**.
- Only user **root** has the write permission on the file that is designated by **path** configured for hooks.
- The value of **timeout** configured for hooks must be greater than **0**.

1.4 Appendix

1.4.1 Command Line Parameters

Table 1-21 login command parameters

Command	Parameter	Description
login	-H, --host	Specifies the iSulad socket file path to be accessed.
	-p, --password	Specifies the password for logging in to the registry.
	--password-stdin	Specifies the password for obtaining the registry from standard input.
	-u, --username	Specifies the username for logging in to the registry.

Table 1-22 logout command parameters

Command	Parameter	Description
logout	-H, --host	Specifies the iSulad socket file path to be accessed.

Table 1-23 pull command parameters

Command	Parameter	Description
pull	-H, --host	Specifies the iSulad socket file path to be accessed.

Table 1-24 rmi command parameters

Command	Parameter	Description
rmi	-H, --host	Specifies the iSulad socket file path to be accessed.
	-f, --force	Forcibly removes an image.

Table 1-25 load command parameters

Command	Parameter	Description
load	-H, --host (supported only by iSula)	Specifies the iSulad socket file path to be accessed.
	-i, --input	Specifies where to import an image. If the image is of the docker type, the value is the image package path. If the image is of the embedded type, the value is the image manifest path.
	--tag	Uses the image name specified by TAG instead of the default image name. This parameter is supported when the type is set to **docker**.
	-t, --type	Specifies the image type. The value can be **embedded** or **docker** (default value).

Table 1-26 images command parameters

Command	Parameter	Description
images	-H, --host	Specifies the iSulad socket file path to be accessed.
	-q, --quit	Displays only the image name.

Table 1-27 inspect command parameters

Command	Parameter	Description
inspect	-H, --host	Specifies the iSulad socket file path to be accessed.
	-f, --format	Outputs using a template.
	-t, --time	Timeout interval, in seconds. If the **inspect** command fails to query container information within the specified period, the system stops waiting and reports an error immediately. The default value is 120s. If the value is less than or equal to 0, the **inspect** command keeps waiting until the container information is obtained successfully.

1.4.2 CNI Parameters

Table 1-28 CNI single network parameters

Parameter	Type	Mandatory or Not	Description
cniVersion	string	Yes	CNI version. Only 0.3.0 and 0.3.1 are supported.
name	string	Yes	Network name, which is user-defined and must be unique.
type	string	Yes	Network type. The following types are supported: underlay_ipvlan overlay_l2 underlay_l2 vpc-router dpdk-direct phy-direct
ipmasp	bool	No	Configures the IP masquerade.
ipam	structure	No	For details, see the IPAM parameter definition.
ipam.type	string	No	IPAM type. The following types are supported: (1) For **underlay_l2**, **overlay_l2**, and **vpc-router** networking, only the default value

Parameter	Type	Mandatory or Not	Description
			distributed_l2 is supported. (2) For **underlay_ipvlan** networking, the default value is **distributed_l2**. In the CCN scenario, only **null** and **fixed** are supported. In the CCE and FST 5G core scenarios, only **null** and **distributed_l2** are supported. (3) For **phy-direct** and **dpdk-direct** networking, the default value is **l2**, and optional values are **null** and **distributed_l2**. In the FST 5G core scenario, only **null** and **distributed_l2** are supported. Description: If the value is out of the range (for example, host-local), Canal automatically sets the value to the default value and no error is returned. **null**: Canal is not used to manage IP addresses. **fixed**: fixed IP address, which is used in the CCN scenario. **l2**: This value is not used in any scenario. **distributed_l2**: The distributed small subnet is used to manage IP addresses.
ipam.subnet	string	No	Subnet information. Canal supports the subnet mask ranging from 8 to 29. The IP address cannot be a multicast address (for example, **224.0.0.0/4**), reserved address (**240.0.0.0/4**), local link address (**169.254.0.0/16**), or local loop address (**127.0.0.0/8**).
ipam.gateway	string	No	Gateway IP address.
ipam.range-start	string	No	Available start IP address.
ipam.range-end	string	No	Available end IP address.
ipam.routes	structure	No	Subnet list. Each element is a route dictionary. For details, see the route definition.
ipam.routes.dst	string	No	Destination network.
ipam.routes.gw	string	No	Gateway address.
dns	structure	No	Contains some special DNS values.
dns.nameservers	[]string	No	NameServers
dns.domain	string	No	Domain
dns.search	[]string	No	Search

Parameter	Type	Mandatory or Not	Description
dns.options	[]string	No	Options
multi_entry	int	No	Number of IP addresses required by a vNIC. The value ranges from 0 to 16. For physical passthrough, a maximum of 128 IP addresses can be applied for a single NIC.
backup_mode	bool	No	Active/Standby mode, which is used only for **phy-direct** and **dpdk-direct** networking.
vlanID	int	No	The value ranges from 0 to 4095. It can be specified through PaaS.
vlan_inside	bool	No	The value **true** indicates that the VLAN function is implemented internally on the node, and the value **false** indicates that the VLAN function is implemented externally.
vxlanID	int	No	The value ranges from 0 to 16777215. It can be specified through PaaS.
vxlan_inside	bool	No	The value **true** indicates that the VLAN function is implemented internally on the node, and the value **false** indicates that the VLAN function is implemented externally.
action	string	No	This parameter can be used only with the special container ID **000000000000**. **Create**: creates a network. **Delete**: deletes a network.
args	map[string]interface{}	No	Key-value pair type.
runtimeConfig	structure	No	None
capabilities	structure	No	None

Table 1-29 CNI args parameters

Parameter	Type	Mandatory	Description
K8S_POD_NAME	string	No	Set this parameter when you apply for a fixed IP address (**runtimeConfig.ican_caps.fixed_ip** is set to **true**).

Parameter	Type	Mandatory	Description
K8S_POD_NAMESPACE	string	No	Set this parameter when you apply for a fixed IP address (**runtimeConfig.ican_caps.fixed_ip** is set to **true**).
SECURE_CONTAINER	string	No	Secure container flag.
multi_port	int	No	The value ranges from 1 to 8. The default value is **1**. Specifies the number of passthrough NICs. Only phy-direct and dpdk-direct networks are supported.
phy-direct	string	No	Specifies the NIC to be connected when you create an SR-IOV container network.
dpdk-direct	string	No	Specifies the NIC to be connected when you create a DPDK passthrough container network.
tenant_id	string	No	Indicates the tenant ID. **Only vpc-router networks are supported.**
vpc_id	string	No	VPC ID. **Only vpc-router networks are supported.**
secret_name	string	No	Specifies the AK/SK object name on the K8S APIServer. **Only vpc-router networks are supported.** For details, see the configuration of VPC-Router logical networks.
IP	string	No	IP address specified by the user, in the format of 192.168.0.10.
K8S_POD_NETWORK_ARGS	string	No	Specifies an IP address, in the format of 192.168.0.10. If both **IP** and **K8S_POD_NETWORK_ARGS** in args are not empty, the value of **K8S_POD_NETWORK_ARGS** prevails.
INSTANCE_NAME	string	No	INSTANCE ID. Refer to fixed IP addresses that support containers.
dist_gateway_disable	bool	No	The value **true** indicates that no gateway is created, and the value **false** indicates that a gateway is created.
phynet	string or []string	No	Specifies the name of the physical plane to be added. The physical plane name is predefined and corresponds to that in the SNC system.

Parameter	Type	Mandatory	Description
			When two plane names are entered, the active and standby planes are supported. Example: phy_net1 or ["phy_net2","phy_net3"]

endpoint_policies	struct	No	"endpoint_policies": [
			{
			"Type": "",
			"ExceptionList": [
			""
			],
			"NeedEncap": true,
			"DestinationPrefix": ""
			}
			]

port_map	struct	No	On a NAT network, container ports can be advertised to host ports. "port_map": [{ "local_port": number, "host_port": number, "protocol": [string...] }...]

Table 1-30 CNI multiple network parameters

Parameter	Type	Mandatory	Description
cniVersion	string	Yes	CNI version. Only 0.3.0 and 0.3.1 are supported.
name	string	Yes	Network name, which is user-defined and must be unique.
plugins	struct	Yes	For details, see Table 1-28

2 System Container

2.1 Overview

System containers are used for heavyweight applications and cloud-based services in scenarios with re-computing, high performance, and high concurrency. Compared with the VM technology, system containers can directly inherit physical machine features and has better performance and less overhead. In addition, system containers can be allocated more computing units of limited resources, reducing costs. Therefore, system containers can be used to build differentiated product competitiveness and provide computing unit instances with higher computing density, lower price, and better performance.

2.2 Installation Guideline

Step 1 Install the container engine iSulad.

```
# yum install iSulad
```

Step 2 Install dependent packages of system containers.

```
# yum install isulad-tools authz isulad-lxcfs-toolkit lxcfs
```

Step 3 Run the following command to check whether iSulad is started:

```
# systemctl status isulad
```

Step 4 Enable the lxcfs and authz services.

```
# systemctl start lxcfs
# systemctl start authz
```

----End

2.3 Usage Guide

2.3.1 Introduction

System container functions are enhanced based on the iSula container engine. The container management function and the command format of the function provided by system containers are the same as those provided by the iSula container engine.

The following sections describe how to use the enhanced functions provided by system containers. For details about other command operations, see 1 iSulad Container Engine.

The system container functions involve only the **isula create/run** command. Unless otherwise specified, this command is used for all functions. The command format is as follows:

```
isula create/run [OPTIONS] [COMMAND] [ARG...]
```

In the preceding format:

- **OPTIONS**: one or more command parameters. For details about supported parameters, see 1 iSulad Container Engine > 1.4 Appendix > 1.4.1 Command Line Parameters.
- **COMMAND**: command executed after a system container is started.
- **ARG**: parameter corresponding to the command executed after a system container is started.

2.3.2 Specifying Rootfs to Create a Container

Function Description

Different from a common container that needs to be started by specifying a container image, a system container is started by specifying a local root file system (rootfs) through the **--external-rootfs** parameter. Rootfs contains the operating system environment on which the container depends during running.

Parameter Description

Command	Parameter	Value Description
isula create/run	--external-rootfs	• Variable of the string type. • Absolute path in the root file system of the container, that is, the path of rootfs.

Constraints

- The rootfs directory specified by the **--external-rootfs** parameter must be an absolute path.
- The rootfs directory specified by the **--external-rootfs** parameter must be a complete OS environment. Otherwise, the container fails to be started.
- When a container is deleted, the rootfs directory specified by **--external-rootfs** is not deleted.

- Containers based on ARM rootfs cannot run on x86 servers. Containers based on x86 rootfs cannot run on ARM servers.
- You are not advised to start multiple container instances by using the same rootfs. That is, one rootfs is used only by container instances in the same lifecycle.

Example

If the local rootfs path is **/root/myrootfs**, run the following command to start a system container:

```
# isula run  -tid --system-container --external-rootfs /root/myrootfs none init
```

NOTE

Rootfs is a user-defined file system. Prepare it by yourself. For example, a rootfs is generated after the TAR package of container images is decompressed.

2.3.3 Using systemd to Start a Container

Function Description

The init process started in system containers differs from that in common containers. Common containers cannot start system services through systemd. However, system containers have this capability. You can enable the systemd service by specifying the **--system-contianer** parameter when starting a system container.

Parameter Description

Command	Parameter	Value Description
isula create/run	--system-container	• The value is of a Boolean data type and can be **true** or **false**. The default value is **true**. • Specifies whether it is a system container. This function must be enabled.

Constraints

- The systemd service needs to call some special system APIs, including mount, umount2, unshare, reboot, and name_to_handle_at. Therefore, permissions to call the preceding APIs are enabled for system containers when the privileged container tag is disabled.
- All system containers are started by the init process. The init process does not respond to the SIGTERM signal which indicates normal exit. By default, the **stop** command forcibly kills the container 10 seconds later. If you need a quicker stop, you can manually specify the timeout duration of the **stop** command.
- **--system-container** must be used together with **--external-rootfs**.
- Various services can run in a system container. The **systemctl** command is used to manage the service starting and stopping. Services may depend on each other. As a result, when an exception occurs, some service processes are in the D or Z state so that the container cannot exit properly.
- Some service processes in a system container may affect other operation results. For example, if the NetworkManager service is running in the container, adding NICs to the

container may be affected (the NICs are successfully added but then stopped by the NetworkManger), resulting in unexpected results.

- Currently, system containers and hosts cannot be isolated by using udev events. Therefore, the **fstab** file cannot be configured.
- The systemd service may conflict with the cgconfig service provided by libcgroup. You are advised to delete the libcgroup-related packages from a container or set **Delegate** of the cgconfig service to **no**.

Example

- Specify the **--system-container** and **--external-rootfs** parameters to start a system container.

```
[root@localhost ~]# isula run -tid -n systest01 --system-container
--external-rootfs /root/myrootfs none init
```

- After the preceding commands are executed, the container is running properly. You can run the **exec** command to access the container and view the process information. The command output indicates that the systemd service has been started.

```
[root@localhost ~]# isula exec -it systest01 bash
[root@localhost /]# ps -ef
UID       PID  PPID  C STIME TTY         TIME CMD
root        1     0  2 06:49 ?       00:00:00 init
root       14     1  2 06:49 ?       00:00:00 /usr/lib/systemd/systemd-journal
root       16     1  0 06:49 ?       00:00:00 /usr/lib/systemd/systemd-network
dbus       23     1  0 06:49 ?       00:00:00 /usr/bin/dbus-daemon --system --
root       25     0  0 06:49 ?       00:00:00 bash
root       59    25  0 06:49 ?       00:00:00 ps -ef
```

- Run the **systemctl** command in the container to check the service status. The command output indicates that the service is managed by systemd.

```
[root@localhost /]# systemctl status dbus
● dbus.service - D-Bus System Message Bus
   Loaded: loaded (/usr/lib/systemd/system/dbus.service; static; vendor preset:
disabled)
   Active: active (running) since Mon 2019-07-22 06:49:38 UTC; 2min 5
8s ago
     Docs: man:dbus-daemon(1)
 Main PID: 23 (dbus-daemon)
   CGroup: /system.slice/dbus.service
           └─23 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidf
ile --systemd-activation --syslog-only

Jul 22 06:49:38 localhost systemd[1]: Started D-Bus System Message Bus.
```

- Run the **systemctl** command in the container to stop or start the service. The command output indicates that the service is managed by systemd.

```
[root@localhost /]# systemctl stop dbus
Warning: Stopping dbus.service, but it can still be activated by:
  dbus.socket
[root@localhost /]# systemctl start dbus
```

2.3.4 Reboot or Shutdown in a Container

Function Description

The **reboot** and **shutdown** commands can be executed in a system container. You can run the **reboot** command to restart a container, and run the **shutdown** command to stop a container.

Parameter Description

Command	Parameter	Value Description
isula create/run	--restart	• Variable of the string type. • Supported option is as follows: **on-reboot**: restarts the system container.

Constraints

- The shutdown function relies on the actual OS of the container running environment.
- When you run the **shutdown -h now** command to shut down the system, do not open multiple consoles. For example, if you run the **isula run -ti** command to open a console and run the **isula attach** command for the container in another host bash, another console is opened. In this case, the **shutdown** command fails to be executed.

Example

- Specify the **--restart on-reboot** parameter when starting a container. For example:

```
[root@localhost ~]# isula run -tid --restart on-reboot --system-container
--external-rootfs /root/myrootfs none init
106faae22a926e22c828a0f2b63cf5c46e5d5986ea8a5b26de81390d0ed9714f
```

- In the container, run the **reboot** command.

```
[root@localhost ~]# isula exec -it 10 bash
[root@localhost /]# reboot
```

Check whether the container is restarted.

```
[root@localhost ~]# isula exec -it 10 ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root        1  0.1  0.0  21588  9504 ?        Ss   12:11   0:00 init
root       14  0.1  0.0  27024  9376 ?        Ss   12:11   0:00 /usr/lib/system
root       17  0.0  0.0  18700  5876 ?        Ss   12:11   0:00 /usr/lib/system
dbus       22  0.0  0.0   9048  3624 ?        Ss   12:11   0:00 /usr/bin/dbus-d
root       26  0.0  0.0   8092  3012 ?        Rs+  12:13   0:00 ps aux
```

- In the container, run the **shutdown** command.

```
[root@localhost ~]# isula exec -it 10 bash
[root@localhost /]# shutdown -h now
[root@localhost /]# [root@localhost ~]#
```

Check whether the container is stopped.

```
[root@localhost ~]# isula exec -it 10 bash
Error response from daemon: Exec container error;Container is not
running:106faae22a926e22c828a0f2b63cf5c46e5d5986ea8a5b26de81390d0ed9714f
```

2.3.5 Configurable Cgroup Path

Function Description

System containers provide the capabilities of isolating and reserving container resources on hosts. You can use the **--cgroup-parent** parameter to specify the cgroup directory used by a container to another directory, thereby flexibly allocating host resources. For example, if the cgroup parent path of containers A, B, and C is set to **/lxc/cgroup1**, and the cgroup parent path of containers D, E, and F is set to **/lxc/cgroup2**, the containers are divided into two groups through the cgroup paths, implementing resource isolation at the cgroup level.

Parameter Description

Command	Parameter	Value Description
isula create/run	--cgroup-parent	• Variable of the string type. • Specifies the cgroup parent path of the container.

In addition to specifying the cgroup parent path for a system container using commands, you can also specify the cgroup paths of all containers by modifying the startup configuration files of the iSulad container engine.

Configuration File Path	Parameter	Description
/etc/isulad/daemon.json	--cgroup-parent	• Variable of the string type. • Specifies the default cgroup parent path of the container. • Example: **"cgroup-parent": "/lxc/mycgroup"**

Constraints

- If the **cgroup parent** parameter is set on both the daemon and client, the value specified on the client takes effect.
- If container A is started before container B, the cgroup parent path of container B is specified as the cgroup path of container A. When deleting a container, you need to delete container B and then container A. Otherwise, residual cgroup resources exist.

Example

Start a system container and specify the **--cgroup-parent** parameter.

```
[root@localhost ~]# isula run -tid --cgroup-parent /lxc/cgroup123 --system-container
--external-rootfs /root/myrootfs none init
115878a4dfc7c5b8c62ef8a4b44f216485422be9a28f447a4b9ecac4609f332e
```

Check the cgroup information of the init process in the container.

```
[root@localhost ~]# isula inspect -f "{{json .State.Pid}}" 11
22167
[root@localhost ~]# cat /proc/22167/cgroup
13:blkio:/lxc/cgroup123/115878a4dfc7c5b8c62ef8a4b44f216485422be9a28f447a4b9ecac460
9f332e
12:perf event:/lxc/cgroup123/115878a4dfc7c5b8c62ef8a4b44f216485422be9a28f447a4b9ec
ac4609f332e
11:cpuset:/lxc/cgroup123/115878a4dfc7c5b8c62ef8a4b44f216485422be9a28f447a4b9ecac46
09f332e
10:pids:/lxc/cgroup123/115878a4dfc7c5b8c62ef8a4b44f216485422be9a28f447a4b9ecac4609
f332e
9:rdma:/lxc/cgroup123/115878a4dfc7c5b8c62ef8a4b44f216485422be9a28f447a4b9ecac4609f
332e
8:devices:/lxc/cgroup123/115878a4dfc7c5b8c62ef8a4b44f216485422be9a28f447a4b9ecac46
09f332e
7:hugetlb:/lxc/cgroup123/115878a4dfc7c5b8c62ef8a4b44f216485422be9a28f447a4b9ecac46
09f332e
6:memory:/lxc/cgroup123/115878a4dfc7c5b8c62ef8a4b44f216485422be9a28f447a4b9ecac460
9f332e
5:net cls,net prio:/lxc/cgroup123/115878a4dfc7c5b8c62ef8a4b44f216485422be9a28f447a
4b9ecac4609f332e
4:cpu,cpuacct:/lxc/cgroup123/115878a4dfc7c5b8c62ef8a4b44f216485422be9a28f447a4b9ec
ac4609f332e
3:files:/lxc/cgroup123/115878a4dfc7c5b8c62ef8a4b44f216485422be9a28f447a4b9ecac4609
f332e
2:freezer:/lxc/cgroup123/115878a4dfc7c5b8c62ef8a4b44f216485422be9a28f447a4b9ecac46
09f332e
1:name=systemd:/lxc/cgroup123/115878a4dfc7c5b8c62ef8a4b44f216485422be9a28f447a4b9e
cac4609f332e/init.scope
0::/lxc/cgroup123/115878a4dfc7c5b8c62ef8a4b44f216485422be9a28f447a4b9ecac4609f332e
```

The cgroup parent path of the container is set to **/sys/fs/cgroup/**<*controller*>**/lxc/cgroup123**.

In addition, you can configure the container daemon file to set the cgroup parent paths for all containers. For example:

```
{
       "cgroup-parent": "/lxc/cgroup123",
}
```

Restart the container engine for the configuration to take effect.

2.3.6 Writable Namespace Kernel Parameters

Function Description

For services running in containers, such as databases, big data, and common applications, some kernel parameters need to be set and adjusted to obtain the optimal performance and reliability. The modification permission of all kernel parameters must be disabled or enabled simultaneously (by using privileged container).

When the modification permission is disabled, only the --sysctl external interface is provided and parameters cannot be flexibly modified in a container.

When the modification permission is enabled, some kernel parameters are globally valid. If some parameters are modified in a container, all programs on the host will be affected, harming security.

System containers provide the **--ns-change-opt** parameter, which can be used to dynamically set namespace kernel parameters in a container. The parameter value can be **net** or **ipc**.

Parameter Description

Command	Parameter	Value Description
isula create/run	--ns-change-opt	• Variable of the string type. • The parameter value can be **net** or **ipc**. **net**: All namespace parameters in the **/proc/sys/net** directory are supported. **ipc**: Supported namespace parameters are as follows: /proc/sys/kernel/msgmax /proc/sys/kernel/msgmnb /proc/sys/kernel/msgmni /proc/sys/kernel/sem /proc/sys/kernel/shmall /proc/sys/kernel/shmmax /proc/sys/kernel/shmmni /proc/sys/kernel/shm_rmid_forced /proc/sys/fs/mqueue/msg_default /proc/sys/fs/mqueue/msg_max /proc/sys/fs/mqueue/msgsize_default /proc/sys/fs/mqueue/msgsize_max /proc/sys/fs/mqueue/queues_max • You can specify multiple namespace configurations and separate them with commas (,). For example, **--ns-change-opt=net,ipc**.

Constraints

- If both **--privileged** (privileged container) and **--ns-change-opt** are specified during container startup, **--ns-change-opt** does not take effect.

Example

Start a container and set **--ns-change-opt** to **net**.

```
[root@localhost ~]# isula run -tid --ns-change-opt net --system-container
--external-rootfs /root/myrootfs none init
4bf44a42b4a14fdaf127616c90defa64b4b532b18efd15b62a71cbf99ebc12d2
[root@localhost ~]# isula exec -it 4b mount | grep /proc/sys
proc on /proc/sys type proc (ro,nosuid,nodev,noexec,relatime)
proc on /proc/sysrq-trigger type proc (ro,nosuid,nodev,noexec,relatime)
proc on /proc/sys/net type proc (rw,nosuid,nodev,noexec,relatime)
```

The mount point **/proc/sys/net** in the container has the **rw** option, indicating that the **net**-related namespace kernel parameters have the read and write permissions.

Start another container and set **--ns-change-opt** to **ipc**.

```
[root@localhost ~]# isula run -tid --ns-change-opt ipc --system-container
--external-rootfs /root/myrootfs none init
c62e5e5686d390500dab2fa76b6c44f5f8da383a4cbbeac12cfada1b07d6c47f
[root@localhost ~]# isula exec -it c6 mount | grep /proc/sys
proc on /proc/sys type proc (ro,nosuid,nodev,noexec,relatime)
proc on /proc/sysrq-trigger type proc (ro,nosuid,nodev,noexec,relatime)
proc on /proc/sys/kernel/shmmax type proc (rw,nosuid,nodev,noexec,relatime)
proc on /proc/sys/kernel/shmmni type proc (rw,nosuid,nodev,noexec,relatime)
proc on /proc/sys/kernel/shmall type proc (rw,nosuid,nodev,noexec,relatime)
proc on /proc/sys/kernel/shm rmid forced type proc (rw,nosuid,nodev,noexec,relatime)
proc on /proc/sys/kernel/msgmax type proc (rw,nosuid,nodev,noexec,relatime)
proc on /proc/sys/kernel/msgmni type proc (rw,nosuid,nodev,noexec,relatime)
proc on /proc/sys/kernel/msgmnb type proc (rw,nosuid,nodev,noexec,relatime)
proc on /proc/sys/kernel/sem type proc (rw,nosuid,nodev,noexec,relatime)
proc on /proc/sys/fs/mqueue type proc (rw,nosuid,nodev,noexec,relatime)
```

The mount point information of **ipc**-related kernel parameters in the container contains the **rw** option, indicating that the **ipc**-related namespace kernel parameters have the read and write permissions.

2.3.7 Shared Memory Channels

Function Description

System containers enable the communication between container and host processes through shared memory. You can set the **--host-channel** parameter when creating a container to allow the host to share the same tmpfs with the container so that they can communicate with each other.

Parameter Description

Command	Parameter	Value Description
isula create/run	--host-channel	• Variable of the string type. Its format is as follows: `<host path>:<container path>:<rw/ro>:<size limit>` • The parameter is described as follows: **<host path>**: path to which tmpfs is mounted on the host, which must be an absolute path. **<container path>**: path to which tmpfs is mounted in a container, which must be an absolute path. **<rw/ro>**: permissions on the file system mounted to

Command	Parameter	Value Description
		the container. The value can only be **rw** (read and write) or **ro** (read only). The default value is **rw**. **<size limit>**: maximum size used by the mounted tmpfs. The minimum value is one 4 KB physical page, and the maximum value is half of the total physical memory in the system. The default value is **64MB**.

Constraints

- The lifecycle of tmpfs mounted on the host starts from the container startup to the container deletion. After a container is deleted and its occupied space is released, the space is removed.
- When a container is deleted, the path to which tmpfs is mounted on the host is deleted. Therefore, an existing directory on the host cannot be used as the mount path.
- To ensure that processes running by non-root users on the host can communicate with containers, the permission for tmpfs mounted on the host is 1777.

Example

Specify the **--host-channel** parameter when creating a container.

```
[root@localhost ~]# isula run --rm -it --host-channel /testdir:/testdir:rw:32M
--system-container --external-rootfs /root/myrootfs none init
root@3b947668eb54:/# dd if=/dev/zero of=/testdir/test.file bs=1024 count=64K
dd: error writing '/testdir/test.file': No space left on device
32769+0 records in
32768+0 records out
33554432 bytes (34 MB, 32 MiB) copied, 0.0766899 s, 438 MB/s
```

NOTE

- If **--host-channel** is used for size limit, the file size is constrained by the memory limit in the container. (The OOM error may occur when the memory usage reaches the upper limit.)
- If a user creates a shared file on the host, the file size is not constrained by the memory limit in the container.
- If you need to create a shared file in the container and the service is memory-intensive, you can add the value of **--host-channel** to the original value of the container memory limit, eliminating the impact.

2.3.8 Dynamically Loading the Kernel Module

Function Description

Services in a container may depend on some kernel modules. You can set environment variables to dynamically load the kernel modules required by services in the container to the host before the system container starts. This feature must be used together with isulad-hooks. For details, see 2.3.12 Dynamically Managing Container Resources (syscontainer-tools).

Parameter Description

Command	Parameter	Value Description
isula create/run	-e KERNEL_MODULES=module_name1,module_name	• Variable of the string type. • This parameter can be set to multiple modules. Use commas (,) to separate module names.

Constraints

- If loaded kernel modules are not verified or conflict with existing modules on the host, an unpredictable error may occur on the host. Therefore, exercise caution when loading kernel modules.
- Dynamic kernel module loading transfers kernel modules to be loaded to containers. This function is implemented by capturing environment variables for container startup using isulad-tools. Therefore, this function relies on the proper installation and deployment of isulad-tools.
- Loaded kernel modules need to be manually deleted.

Example

When starting a system container, specify the **-e KERNEL_MODULES** parameter. After the system container is started, the ip_vs module is successfully loaded to the kernel.

```
[root@localhost ~]# lsmod | grep ip vs
[root@localhost ~]# isula run -tid -e KERNEL MODULES=ip vs,ip vs wrr --hook-spec
/etc/isulad-tools/hookspec.json --system-container --external-rootfs /root/myrootfs
none init
ae18c4281d5755a1e153a7bff6b3b4881f36c8e528b9baba8a3278416a5d0980
[root@localhost ~]# lsmod | grep ip vs
ip vs wrr            16384  0
ip vs               176128  2 ip vs wrr
nf conntrack         172032  7
xt conntrack,nf nat,nf nat ipv6,ipt MASQUERADE,nf nat ipv4,nf conntrack netlink,ip
 vs
nf defrag ipv6        20480  2 nf conntrack,ip vs
libcrc32c            16384  3 nf_conntrack,nf_nat,ip_vs
```

NOTE

- isulad-tools must be installed on the host.
- **--hooks-spec** must be set to **isulad hooks**.

2.3.9 Environment Variable Persisting

Function Description

In a system container, you can make the **env** variable persistent to the configuration file in the rootfs directory of the container by specifying the **--env-target-file** interface parameter.

Parameter Description

Command	Parameter	Value Description
isula create/run	--env-target-file	• Variable of the string type. • The **env** persistent file must be in the rootfs directory and must be an absolute path.

Constraints

- If the target file specified by **--env-target-file** exists, the size cannot exceed 10 MB.
- The parameter specified by **--env-target-file** must be an absolute path in the rootfs directory.
- If the value of **--env** conflicts with that of **env** in the target file, the value of **--env** prevails.

Example

Start a system container and specify the **env** environment variable and **--env-target-file** parameter.

```
[root@localhost ~]# isula run -tid -e abc=123 --env-target-file /etc/environment
--system-container --external-rootfs /root/myrootfs none init
b75df997a64da74518deb9a01d345e8df13eca6bcc36d6fe40c3e90ea1ee088e
[root@localhost ~]# isula exec b7 cat /etc/environment
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
TERM=xterm
abc=123
```

The preceding information indicates that the **env** variable (**abc=123**) of the container has been made persistent to the **/etc/environment** configuration file.

2.3.10 Maximum Number of Handles

Function Description

System containers support limit on the number of file handles. File handles include common file handles and network sockets. When starting a container, you can specify the **--files-limit** parameter to limit the maximum number of handles opened in the container.

Parameter Description

Command	Parameter	Value Description

Command	Parameter	Value Description
isula create/run	--files-limit	• The value cannot be negative and must be an integer. • The value **0** indicates that the number is not limited by the parameter. The maximum number is determined by the current kernel files cgroup.

Constraints

- If the value of **--files-limit** is too small, the system container may fail to run the **exec** command and the error "open temporary files" is reported. Therefore, you are advised to set the parameter to a large value.
- File handles include common file handles and network sockets.

Example

To use **--files-limit** to limit the number of file handles opened in a container, run the following command to check whether the kernel supports files cgroup:

```
[root@localhost ~]# cat /proc/1/cgroup | grep files
10:files:/
```

If **files** is displayed, files cgroup is supported.

Start the container, specify the **--files-limit** parameter, and check whether the **files.limit** parameter is successfully written.

```
[root@localhost ~]# isula run  -tid --files-limit 1024 --system-container
--external-rootfs /tmp/root-fs  empty init
01e82fcf97d4937aa1d96eb8067f9f23e4707b92de152328c3fc0ecb5f64e91d
[root@localhost ~]# isula exec -it 01e82fcf97d4 bash
[root@localhost ~]# cat /sys/fs/cgroup/files/files.limit
1024
```

The preceding information indicates that the number of file handles is successfully limited in the container.

2.3.11 Security and Isolation

2.3.11.1 Many-to-Many User Namespaces

Function Description

User namespaces are used to map user **root** of a container to a common user of the host and allow the processes and user in the container (that are unprivileged on the host) to have privilege. This can prevent the processes in the container from escaping to the host and performing unauthorized operations. In addition, after user namespaces are used, the container and host use different UIDs and GIDs. This ensures that user resources in the container such as file descriptors are isolated from those on the host.

In system containers, you can configure the **--user-remap** API parameter to map user namespaces of different containers to different user namespaces on the host, isolating the user namespaces of containers.

Parameter Description

Command	Parameter	Value Description
isula create/run	--user-remap	The parameter format is *uid*:*gid*:*offset*. The parameter is described as follows: • *uid* and *gid* must be integers greater than or equal to 0. • *offset* must be an integer greater than 0 and less than 65536. The value cannot be too small. Otherwise, the container cannot be started. • Either the sum of *uid* and *offset* or the sum of *gid* and *offset* must be less than or equal to 2^{32} - 1. Otherwise, an error is reported during container startup.

Constraints

- If **--user-remap** is specified in a system container, the rootfs directory must be accessible to users specified by *uid* or *gid* in **--user-remap**. Otherwise, user namespaces of containers cannot access rootfs. As a result, the containers fail to be started.
- All IDs in the container can be mapped to the host rootfs. Some directories or files may be mounted from the host to containers, for example, device files in the **/dev/pts** directory. If *offset* is too small, the mounting may fail.
- *uid*, *gid*, and *offset* are controlled by the upper-layer scheduling platform. The container engine only checks the validity of them.
- **--user-remap** is available only in system containers.
- **--user-remap** and **--privileged** cannot be set simultaneously. Otherwise, an error is reported during container startup.
- If *uid* or *gid* is set to **0**, **--user-remap** does not take effect.

Usage Guide

NOTE

Before specifying the **--user-remap** parameter, configure an offset value for UIDs and GIDs of all directories and files in rootfs. The offset value should be equal to that for *uid* and *gid* in **--user-remap**.

For example, run the following command to offset UIDs and GIDs of all files in the **dev** directory with 100000:

chown 100000:100000 dev

Specify the **--user-remap** parameter when the system container is started.

```
[root@localhost ~]# isula run -tid --user-remap 100000:100000:65535 --system-container
--external-rootfs /home/root-fs none /sbin/init
eb9605b3b56dfae9e0b696a729d5e1805af900af6ce24428fde63f3b0a443f4a
```

Check the /sbin/init process information on the host and in a container.

```
[root@localhost ~]# isula exec eb ps aux | grep /sbin/init
root        1  0.6  0.0  21624  9624 ?       Ss   15:47   0:00 /sbin/init
[root@localhost ~]# ps aux | grep /sbin/init
100000    4861  0.5  0.0  21624  9624 ?       Ss   15:47   0:00 /sbin/init
root      4948  0.0  0.0 213032   808 pts/0    S+   15:48   0:00 grep --color=auto
/sbin/init
```

The owner of the /sbin/init process in the container is user **root**, but the owner of the host is the user whose UID is **100000**.

Create a file in a container and view the file owner on the host.

```
[root@localhost ~]# isula exec -it eb bash
[root@localhost /]# echo test123 >> /test123
[root@localhost /]# exit
exit
[root@localhost ~]# ll /home/root-fs/test123
-rw-------. 1 100000 100000 8 Aug  2 15:52 /home/root-fs/test123
```

The owner of the file that is generated in the container is user **root**, but the file owner displayed on the host is the user whose ID is **100000**.

2.3.11.2 User Permission Control

Function Description

A container engine supports TLS for user identity authentication, which is used to control user permissions. Currently, container engines can connect to the authz plug-in to implement permission control.

API Description

You can configure the startup parameters of the iSulad container engine to specify the permission control plug-in. The default daemon configuration file is **/etc/isulad/daemon.json**.

Parameter	Example	Description
--authorization-plugin	"authorization-plugin": "authz-broker"	User permission authentication plug-in. Currently, only authz-broker is supported.

Constraints

- User permission policies need to be configured for authz. The default policy file is **/var/lib/authz-broker/policy.json**. This file can be dynamically modified and the modification will take effect immediately without restarting the plug-in service.
- A container engine can be started by user **root**. If some commands used are enabled for by common users, common users may obtain excessive permissions. Therefore, exercise caution when performing such operations. Currently, running the **container_attach**, **container_create**, and **container_exec_create** commands may cause risks.
- Some compound operations, such as running **isula exec** and **isula inspect** or running and **isula attach** and **isula inspect**, depend on the permission of **isula inspect**. If a user does not have this permission, an error is reported.

- Using SSL/TLS encryption channels hardens security but also reduces performance. For example, the delay increases, more CPU resources are consumed, and encryption and decryption require higher throughput. Therefore, the number of concurrent executions decreases compared with non-TLS communication. According to the test result, when the ARM server (Cortex-A72 64-core) is almost unloaded, TLS is used to concurrently start a container. The maximum number of concurrent executions is 200 to 250.
- If **--tlsverify** is specified on the server, the default path where authentication files store is **/etc/isulad**. The default file names are **ca.pem**, **cert.pem**, and **key.pem**.

Example

Step 1 Ensure that the authz plug-in is installed on the host. If the authz plug-in is not installed, run the following command to install and start the authz plug-in service:

```
[root@localhost ~]# yum install authz
[root@localhost ~]# systemctl start authz
```

Step 2 To enable this function, configure the container engine and TLS certificate. You can use OpenSSL to generate the required certificate.

```
#SERVERSIDE

# Generate CA key
openssl genrsa -aes256 -passout "pass:$PASSWORD" -out "ca-key.pem" 4096
# Generate CA
openssl req -new -x509 -days $VALIDITY -key "ca-key.pem" -sha256 -out "ca.pem" -passin
"pass:$PASSWORD" -subj
"/C=$COUNTRY/ST=$STATE/L=$CITY/O=$ORGANIZATION/OU=$ORGANIZATIONAL UNIT/CN=$COMMON
NAME/emailAddress=$EMAIL"
# Generate Server key
openssl genrsa -out "server-key.pem" 4096

# Generate Server Certs.
openssl req -subj "/CN=$COMMON NAME" -sha256 -new -key "server-key.pem" -out server.csr

echo "subjectAltName = DNS:localhost,IP:127.0.0.1" > extfile.cnf
echo "extendedKeyUsage = serverAuth" >> extfile.cnf

openssl x509 -req -days $VALIDITY -sha256 -in server.csr -passin "pass:$PASSWORD" -CA
"ca.pem" -CAkey "ca-key.pem" -CAcreateserial -out "server-cert.pem" -extfile
extfile.cnf

#CLIENTSIDE

openssl genrsa -out "key.pem" 4096
openssl req -subj "/CN=$CLIENT NAME" -new -key "key.pem" -out client.csr
echo "extendedKeyUsage = clientAuth" > extfile.cnf
openssl x509 -req -days $VALIDITY -sha256 -in client.csr -passin "pass:$PASSWORD" -CA
"ca.pem" -CAkey "ca-key.pem" -CAcreateserial -out "cert.pem" -extfile extfile.cnf
```

If you want to use the preceding content as the script, replace the variables with the configured values. If the parameter used for generating the CA is empty, set it to ''. **PASSWORD**, **COMMON_NAME**, **CLIENT_NAME**, and **VALIDITY** are mandatory.

Step 3 When starting the container engine, add parameters related to the TLS and authentication plug-in and ensure that the authentication plug-in is running properly. In addition, to use TLS

authentication, the container engine must be started in TCP listening mode instead of the Unix socket mode. The configuration on the container demon is as follows:

```
{
   "tls": true,
   "tls-verify": true,
   "tls-config": {
         "CAFile": "/root/.iSulad/ca.pem",
         "CertFile": "/root/.iSulad/server-cert.pem",
         "KeyFile":"/root/.iSulad/server-key.pem"
   },
   "authorization-plugin": "authz-broker"
}
```

Step 4 Configure policies. For the basic authorization process, all policies are stored in the **/var/lib/authz-broker/policy.json** configuration file. The configuration file can be dynamically modified without restarting the plug-in. Only the SIGHUP signal needs to be sent to the authz process. In the file, a line contains one JSON policy object. The following provides policy configuration examples:

- All users can run all iSuald commands:
  {"name":"policy_0","users":[""],"actions":[""]}
- Alice can run all iSulad commands:
  {"name":"policy_1","users":["alice"],"actions":[""]}
- A blank user can run all iSulad commands:
  {"name":"policy_2","users":[""],"actions":[""]}
- Alice and Bob can create new containers:
  {"name":"policy_3","users":["alice","bob"],"actions":["container_create"]}
- service_account can read logs and run **docker top**:
  {"name":"policy_4","users":["service_account"],"actions":["container_logs","container_top"]}
- Alice can perform any container operations:
  {"name":"policy_5","users":["alice"],"actions":["container"]}
- Alice can perform any container operations, but the request type can only be **get**:
  {"name":"policy_5","users":["alice"],"actions":["container"], "readonly":true}

NOTE

- **action** indicates that regular expressions are supported.
- **users** indicates that regular expressions are not supported.
- Users configured in **users** must be unique. That is, a user cannot match multiple rules.

Step 5 After updating the configurations, configure TLS parameters on the client to connect to the container engine. That is, access the container engine with restricted permissions.

```
[root@localhost ~]#  isula version --tlsverify --tlscacert=/root/.iSulad/ca.pem
--tlscert=/root/.iSulad/cert.pem --tlskey=/root/.iSulad/key.pem
-H=tcp://127.0.0.1:2375
```

If you want to use the TLS authentication for default client connection, move the configuration file to **~/.iSulad** and set the **ISULAD_HOST** and **ISULAD_TLS_VERIFY** variables (rather than transferring **-H=tcp://$HOST:2375** and **--tlsverify** during each call).

```
[root@localhost ~]# mkdir -pv ~/.iSulad
[root@localhost ~]# cp -v {ca,cert,key}.pem ~/.iSulad
```

```
[root@localhost ~]# export ISULAD_HOST=localhost:2375 ISULAD_TLS_VERIFY=1
[root@localhost ~]# isula version
```

----End

2.3.11.3 proc File System Isolation (Lxcfs)

Application Scenario

Container virtualization is lightweight and efficient, and can be quickly deployed. However, containers are not strongly isolated, which causes great inconvenience to users. Containers have some defects in isolation because the namespace feature of the Linux kernel is not perfect. For example, you can view the proc information on the host (such as meminfo, cpuinfo, stat, and uptime) in the proc file system of a container. You can use the lxcfs tool to replace the /proc content of instances in the container with the content in the /proc file system of the host so that services in the container can obtain the correct resource value.

API Description

A system container provides two tool packages: lxcfs and lxcfs-toolkit, which are used together. Lxcfs resides on the host as the daemon process. lxcfs-toolkit mounts the lxcfs file system of the host to containers through the hook mechanism.

The command line of lxcfs-toolkit is as follows:

```
lxcfs-toolkit [OPTIONS] COMMAND [COMMAND OPTIONS]
```

Command	Function	Parameter
remount	Remounts lxcfs to containers.	**--all**: remounts lxcfs to all containers. **--container-id**: remounts lxcfs to a specified container.
umount	Unmounts lxcfs from containers.	**--all**: unmounts lxcfs from all containers. **--container-id**: unmounts lxcfs from a specified container.
check-lxcfs	Checks whether the lxcfs service is running properly.	None
prestart	Mounts the **/var/lib/lxcfs** directory to the container before the lxcfs service starts.	None

Constraints

- Currently, only the **cpuinfo**, **meminfo**, **stat**, **diskstats**, **partitions**, **swaps**, and **uptime** files in the proc file system are supported. Other files are not isolated from other kernel API file systems (such as sysfs).
- After an RPM package is installed, a sample JSON file is generated in **/var/lib/lcrd/hooks/hookspec.json**. To add the log function, you need to add the **--log** configuration during customization.

- The **diskstats** file displays only information about disks that support CFQ scheduling, instead of partition information. Devices in containers are displayed as names in the **/dev** directory. If a device name does not exist, the information is left blank. In addition, the device where the container root directory is located is displayed as **sda**.
- The **slave** parameter is required when lxcfs is mounted. If the **shared** parameter is used, the mount point in containers may be leaked to the host, affecting the host running.
- Lxcfs supports graceful service degradation. If the lxcfs service crashes or becomes unavailable, the **cpuinfo**, **meminfo**, **stat**, **diskstats**, **partitions**, **swaps** and **uptime** files in containers are about host information, and other service functions of containers are not affected.
- Bottom layer of lxcfs depends on the FUSE kernel module and libfuse library. Therefore, the kernel needs to support FUSE.
- Lxcfs supports only the running of 64-bit applications in containers. If a 32-bit application is running in a container, the CPU information (**cpuinfo**) read by the application may fail to meet expectations.
- Lxcfs simulates the resource view only of container control groups (cgroups). Therefore, system calls (such as sysconf) in containers can obtain only host information. Lxcfs cannot implement the kernel isolation.
- The CPU information (**cpuinfo**) displayed after lxcfs implements the isolation has the following features:
  - **processor**: The value increases from 0.
  - **physical id**: The value increases from 0.
  - **sibliing**: It has a fixed value of **1**.
  - **core id**: It has a fixed value of **0**.
  - **cpu cores**: It has a fixed value of **1**.

Example

Step 1 Install the lxcfs and lxcfs-toolkit packages and start the lxcfs service.

```
[root@localhost ~]# yum install lxcfs lxcfs-toolkit
[root@localhost ~]# systemctl start lxcfs
```

Step 2 After a container is started, check whether the lxcfs mount point exists in the container.

```
[root@localhost ~]# isula run -tid -v /var/lib/lxc:/var/lib/lxc --hook-spec
/var/lib/isulad/hooks/hookspec.json --system-container --external-rootfs
/home/root-fs none init
a8acea9fea1337d9fd8270f41c1a3de5bceb77966e03751346576716eefa9782
[root@localhost ~]# isula exec a8 mount | grep lxcfs
lxcfs on /var/lib/lxc/lxcfs type fuse.lxcfs
(rw,nosuid,nodev,relatime,user id=0,group id=0,allow other)
lxcfs on /proc/cpuinfo type fuse.lxcfs
(rw,nosuid,nodev,relatime,user id=0,group id=0,allow other)
lxcfs on /proc/diskstats type fuse.lxcfs
(rw,nosuid,nodev,relatime,user id=0,group id=0,allow other)
lxcfs on /proc/meminfo type fuse.lxcfs
(rw,nosuid,nodev,relatime,user id=0,group id=0,allow other)
lxcfs on /proc/partitions type fuse.lxcfs
(rw,nosuid,nodev,relatime,user id=0,group id=0,allow other)
lxcfs on /proc/stat type fuse.lxcfs
(rw,nosuid,nodev,relatime,user_id=0,group_id=0,allow_other)
```

```
lxcfs on /proc/swaps type fuse.lxcfs
(rw,nosuid,nodev,relatime,user id=0,group id=0,allow other)
lxcfs on /proc/uptime type fuse.lxcfs
(rw,nosuid,nodev,relatime,user_id=0,group_id=0,allow_other)
```

Step 3 Run the **update** command to update the CPU and memory resource configurations of the container and check the container resources. As shown in the following command output, the container resource view displays the actual container resource data instead of data of the host.

```
[root@localhost ~]# isula update --cpuset-cpus 0-1 --memory 1G a8
a8
[root@localhost ~]# isula exec a8 cat /proc/cpuinfo
processor      : 0
BogoMIPS       : 100.00
cpu MHz        : 2400.000
Features       : fp asimd evtstrm aes pmull sha1 sha2 crc32 cpuid
CPU implementer : 0x41
CPU architecture: 8
CPU variant    : 0x0
CPU part       : 0xd08
CPU revision   : 2

processor      : 1
BogoMIPS       : 100.00
cpu MHz        : 2400.000
Features       : fp asimd evtstrm aes pmull sha1 sha2 crc32 cpuid
CPU implementer : 0x41
CPU architecture: 8
CPU variant    : 0x0
CPU part       : 0xd08
CPU revision   : 2

[root@localhost ~]# isula exec a8 free -m
              total        used        free      shared  buff/cache   available
Mem:           1024          17         997           7           8        1006
Swap:          4095           0        4095
```

----End

2.3.12 Dynamically Managing Container Resources (syscontainer-tools)

Resources in common containers cannot be managed. For example, a block device cannot be added to a common container, and a physical or virtual NIC cannot be inserted to a common container. In the system container scenario, the syscontainer-tools can be used to dynamically mount or unmount block devices, network devices, routes, and volumes for containers.

To use this function, you need to install the syscontainer-tools first.

```
[root@localhost ~]# yum install syscontainer-tools
```

2.3.12.1 Device Management

Function Description

isulad-tools allows you to add block devices (such as disks and logical volume managers) or character devices (such as GPUs, binners, and FUSEs) on the host to a container. The devices can be used in the container. For example, you can run the **fdisk** command to format the disk and write data to the file system. If the devices are not required, isulad-tools allows you to delete them from the container and return them to the host.

Command Format

```
isulad-tools [COMMADN][OPTIONS] <container_id> [ARG...]
```

In the preceding format:

COMMAND: command related to device management.

OPTIONS: option supported by the device management command.

container_id: container ID.

ARG: parameter corresponding to the command.

Parameter Description

Command	Function Description	Option Description	Parameter Description
add-device	Adds block devices or character devices on the host to a container.	Supported options are as follows: • **--blkio-weight-device**: sets the I/O weight (relative weight, ranging from 10 to 100) of a block device. • **--device-read-bps**: sets the read rate limit for the block device (byte/s). • **--device-read-iops**: sets the read rate limit for the block device (I/O/s). • **--device-write-bps**: sets the write rate limit for the block device (byte/s). • **--device-write-iops**: sets the write rate limit for the block device (I/O/s). • **--follow-partition**: If a block device is a basic block device (primary SCSI block disk), set this parameter to add all partitions of the primary	Parameter format: *hostdevice***[:***containerdevice***][:***permission***][***hostdevice***[:***containerdevice***][:***permission***]]** In the preceding format: *hostdevice*: path on the host for storing a device. *containerdevice*: path on the container for storing a device. *permission*: operation permission on a device within the container.

Command	Function Description	Option Description	Parameter Description
		disk. • **--force**: If any block device or character device already exists in the container, use this parameter to overwrite the old block device or character device files. • **--update-config-only**: updates configuration files only and does not add disks.	
remove-device	Deletes block devices or character devices from a container and restores them to the host.	Supported options are as follows: **--follow-partition**: If a block device is a basic block device (primary SCSI block disk), set this parameter to delete all partitions of the primary disk in the container, and restore them to the host.	Parameter format: *hostdevice***[:***containerdevice***] [***hostdevice***[:***containerdevice***]]** In the preceding format: *hostdevice*: path on the host for storing a device. *containerdevice*: path on the container for storing a device.
list-device	Lists all block devices or character devices in a container.	Supported options are as follows: • **--pretty**: outputs data in JSON format. • **--sub-partition**: For a primary disk, add this flag to display the primary disk and its sub-partitions.	None
update-device	Updates the disk QoS.	Supported options are as follows: • **--device-read-bps**: sets the read rate limit for the block device (byte/s). You are advised to set this parameter to a value greater than or equal to 1024. • **--device-read-iops**: sets the read rate limit for the block device (I/O/s). • **--device-write-bps**: sets the write rate limit for the block device (byte/s).	None

Command	Function Description	Option Description	Parameter Description
		You are advised to set this parameter to a value greater than or equal to 1024. • **--device-write-iops**: sets the write rate limit for the block device (I/O/s).	

Constraints

- You can add or delete devices when container instances are not running. After the operation is complete, you can start the container to view the device status. You can also dynamically add a device when the container is running.
- Do not concurrently run the **fdisk** command to format disks in a container and on the host. Otherwise, the container disk usage will be affected.
- When you run the **add-device** command to add a disk to a specific directory of a container, if the parent directory in the container is a multi-level directory (for example, **/dev/a/b/c/d/e**) and the directory level does not exist, isulad-tools will automatically create the corresponding directory in the container. When the disk is deleted, the created parent directory is not deleted. If you run the **add-device** command to add a device to this parent directory again, a message is displayed, indicating that a device already exists and cannot be added.
- When you run the **add-device** command to add a disk or update disk parameters, you need to configure the disk QoS. Do not set the write or read rate limit for the block device (I/O/s or byte/s) to a small value. If the value is too small, the disk may be unreadable (the actual reason is the speed is too slow), affecting service functions.
- When you run the **--blkio-weight-device** command to limit the weight of a specified block device, if the block device supports only the BFQ mode, an error may be reported, prompting you to check whether the current OS environment supports setting the weight of the BFQ block device.

Example

- Start a system container, and set **hook spec** to the isulad hook execution script.

```
[root@localhost ~]# isula run -tid --hook-spec /etc/isulad-tools/hookspec.json
--system-container --external-rootfs /root/root-fs none init
eed1096c8c7a0eca6d92b1b3bc3dd59a2a2adf4ce44f18f5372408ced88f8350
```

- Add a block device to a container.

```
[root@localhost ~]# isulad-tools add-device ee /dev/sdb:/dev/sdb123
Add device (/dev/sdb) to container(ee,/dev/sdb123) done.
[root@localhost ~]# isula exec ee fdisk -l /dev/sdb123
Disk /dev/sdb123: 50 GiB, 53687091200 bytes, 104857600 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xda58a448
```

```
Device       Boot Start      End   Sectors Size Id Type
/dev/sdb123p1      2048 104857599 104855552  50G  5 Extended
/dev/sdb123p5      4096 104857599 104853504  50G 83 Linux
```

- Update the device information.

```
[root@localhost ~]# isulad-tools update-device --device-read-bps /dev/sdb:10m ee
Update read bps for device (/dev/sdb,10485760) done.
```

- Delete a device.

```
[root@localhost ~]# isulad-tools remove-device ee /dev/sdb:/dev/sdb123
Remove device (/dev/sdb) from container(ee,/dev/sdb123) done.
Remove read bps for device (/dev/sdb) done.
```

2.3.12.2 NIC Management

Function Description

isulad-tools allows you to insert physical or virtual NICs on the host to a container. If the NICs are not required, isulad-tools allows you to delete them from the container and return them to the host. In addition, the NIC configurations can be dynamically modified. To insert a physical NIC, add the NIC on the host to the container. To insert a virtual NIC, create a veth pair and insert its one end to the container.

Command Format

```
isulad-tools [COMMADN][OPTIONS] <container_id>
```

In the preceding format:

COMMAND: command related to NIC management.

OPTIONS: option supported by the NIC management command.

container_id: container ID.

Parameter Description

Command	Function Description	Option Description
add-nic	Creates an NIC for a container.	Supported options are as follows: • **--type**: specifies the NIC type. Only **eth** and **veth** are supported. • **--name**: specifies the NIC name. The format is **[***host***:]***container*. If *host* is not specified, a random value is used. • **--ip**: specifies the NIC IP address. • **--mac**: specifies the NIC MAC address. • **--bridge**: specifies the network bridge bound to the NIC. • **--mtu**: specifies the MTU value of the NIC. The default value is **1500**. • **--update-config-only**: If this flag is set, only configuration files are

Command	Function Description	Option Description
		updated and NICs are not added. • **--qlen**: specifies the value of QLEN. The default value is **1000**.
remove-nic	Deletes NICs from a container and restores them to the host.	Supported options are as follows: • **--type**: specifies the NIC type. • **--name**: specifies the name of the NIC. The format is **[***host***:]***container*.
list-nic	Lists all NICs in a container.	Supported options are as follows: • **--pretty**: outputs data in JSON format. • **--filter**: outputs filtered data in the specific format, for example, **--filter' {"ip":"192.168.3.4/24", "Mtu":1500}**'.
update-nic	Modifies configuration parameters of a specified NIC in a container.	Supported options are as follows: • **--name**: specifies the name of the NIC in the container. This parameter is mandatory. • **--ip**: specifies the NIC IP address. • **--mac**: specifies the NIC MAC address. • **--bridge**: specifies the network bridge bound to the NIC. • **--mtu**: specifies the MTU value of the NIC. • **--update-config-only**: If this flag is set, configuration files are updated and NICs are not updated. • **--qlen**: specifies the value of QLEN.

Constraints

- Physical NICs (eth) and virtual NICs (veth) can be added.
- When adding a NIC, you can also configure the NIC. The configuration parameters include **--ip**, **--mac**, **--bridge**, **--mtu**, **--qlen**.
- A maximum of eight physical NICs can be added to a container.
- If you run the **isulad-tools add-nic** command to add an eth NIC to a container and do not add a hook, you must manually delete the NIC before the container exits. Otherwise, the name of the eth NIC on the host will be changed to the name of that in the container.
- For a physical NIC (except 1822 VF NIC), use the original MAC address when running the **add-nic** command. Do not change the MAC address in the container, or when running the **update-nic** command.
- When using the **isulad-tools add-nic** command, set the MTU value. The value range depends on the NIC model.

- When using isulad-tools to add NICs and routes to containers, you are advised to run the **add-nic** command to add NICs and then run the **add-route** command to add routes. When using isulad-tools to delete NICs and routes from a container, you are advised to run the **remove-route** command to delete routes and then run the **remove-nic** command to delete NICs.
- When using isulad-tools to add NICs, add a NIC to only one container.

Example

- Start a system container, and set **hook spec** to the isulad hook execution script.

```
[root@localhost ~]# isula run -tid --hook-spec /etc/isulad-tools/hookspec.json
--system-container --external-rootfs /root/root-fs none init
2aaca5c1af7c872798dac1a468528a2ccbaf20b39b73fc0201636936a3c32aa8
```

- Add a virtual NIC to a container.

```
[root@localhost ~]# isulad-tools add-nic --type "veth" --name abc2:bcd2 --ip
172.17.28.5/24 --mac 00:ff:48:13:xx:xx --bridge docker0 2aaca5c1af7c
Add network interface to container 2aaca5c1af7c (bcd2,abc2) done
```

- Add a physical NIC to a container.

```
[root@localhost ~]# isulad-tools add-nic --type "eth" --name eth3:eth1 --ip
172.17.28.6/24  --mtu 1300  --qlen 2100 2aaca5c1af7c
Add network interface to container 2aaca5c1af7c (eth3,eth1) done
```

NOTE

When adding a virtual or physical NIC, ensure that the NIC is in the idle state. Adding a NIC in use will disconnect the system network.

2.3.12.3 Route Management

Function Description

isulad-tools can be used to dynamically add or delete routing tables for system containers.

Command Format

```
isulad-tools [COMMADN][OPTIONS] <container_id> [ARG...]
```

In the preceding format:

COMMAND: command related to route management.

OPTIONS: option supported by the route management command.

container_id: container ID.

ARG: parameter corresponding to the command.

API Description

Command	Function Description	Option Description	Parameter Description
add-route	Adds the network routing rules to a container.	Supported options are as follows: **--update-config-only**: If	Parameter format: **[{***rule1***},{***rule2***}]** Example of *rule*:

Command	Function Description	Option Description	Parameter Description
		this parameter is configured, configuration files are updated and routing tables are not updated.	'[{"dest":"default", "gw":"192.168.10.1"},{"dest":"192.168.0.0/16","dev":"eth0","src":"192.168.1.2"}]' • **dest**: target network. If this parameter is left blank, the default gateway is used. • **src**: source IP address of a route. • **gw**: route gateway. • **dev**: network device.
remove-route	Deletes a route from a container.	Supported options are as follows: **--update-config-only**: If this parameter is configured, only configuration files are updated and routes are not deleted from the container.	Parameter format: **[***{rule1}***,***{rule2}***]** Example of *rule*: '[{"dest":"default", "gw":"192.168.10.1"},{"dest":"192.168.0.0/16","dev":"eth0","src":"192.168.1.2"}]' • **dest**: target network. If this parameter is left blank, the default gateway is used. • **src**: source IP address of a route. • **gw**: route gateway. • **dev**: network device.
list-route	Lists all routing rules in a container.	Supported options are as follows: • **--pretty**: outputs data in JSON format. • **--filter**: outputs filtered data in the specific format, for example, **--filter' {"ip":"192.168.3.4/24", "Mtu":1500}**'.	None

Constraints

- When using isulad-tools to add NICs and routes to containers, you are advised to run the **add-nic** command to add NICs and then run the **add-route** command to add routes. When using isulad-tools to delete NICs and routes from a container, you are advised to

run the **remove-route** command to delete routes and then run the **remove-nic** command to delete NICs.

- When adding a routing rule to a container, ensure that the added routing rule does not conflict with existing routing rules in the container.

Example

- Start a system container, and set **hook spec** to the isulad hook execution script.

```
[root@localhost ~]# isula run -tid --hook-spec /etc/isulad-tools/hookspec.json
--system-container --external-rootfs /root/root-fs none init
0d2d68b45aa0c1b8eaf890c06ab2d008eb8c5d91e78b1f8fe4d37b86fd2c190b
```

- Use isulad-tools to add a physical NIC to the system container.

```
[root@localhost ~]# isulad-tools add-nic --type "eth" --name enp4s0:eth123 --ip
172.17.28.6/24  --mtu 1300  --qlen 2100 0d2d68b45aa0
Add network interface (enp4s0) to container (0d2d68b45aa0,eth123) done
```

- isulad-tools adds a routing rule to the system container. Format example: **[{"dest":"default", "gw":"192.168.10.1"},{"dest":"192.168.0.0/16","dev":"eth0","src":"192.168.1.2"}]**. If **dest** is left blank, its value will be **default**.

```
[root@localhost ~]# isulad-tools add-route 0d2d68b45aa0
'[{"dest":"172.17.28.0/32", "gw":"172.17.28.5","dev":"eth123"}]'
Add route to container 0d2d68b45aa0, route:
{dest:172.17.28.0/32,src:,gw:172.17.28.5,dev:eth123} done
```

- Check whether a routing rule is added in the container.

```
[root@localhost ~]# isula exec -it 0d2d68b45aa0 route
Kernel IP routing table
Destination    Gateway        Genmask        Flags Metric Ref    Use Iface
172.17.28.0    172.17.28.5     255.255.255.255 UGH   0      0        0 eth123
172.17.28.0    0.0.0.0        255.255.255.0   U     0      0        0 eth123
```

2.3.12.4 Volume Mounting Management

Function Description

In a common container, you can set the **--volume** parameter during container creation to mount directories or volumes of the host to the container for resource sharing. However, during container running, you cannot unmount directories or volumes that are mounted to the container, or mount directories or volumes of the host to the container. Only the system container can use the isulad-tools tool to dynamically mount directories or volumes of the host to the container and unmount directories or volumes from the container.

Command Format

```
isulad-tools [COMMADN][OPTIONS] <container_id> [ARG...]
```

In the preceding format:

COMMAND: command related to route management.

OPTIONS: option supported by the route management command.

container_id: container ID.

ARG: parameter corresponding to the command.

API Description

Table 2-1

Command	Function Description	Option Description	Parameter Description
add-path	Adds files or directories on the host to a container.	None	The parameter format is as follows: hostpath:containerpath:permission [hostpath:containerpath:permission ...] In the preceding format: *hostpath*: path on the host for storing a volume. *containerpath*: path on the container for storing a volume. *permission*: operation permission on a mount path within the container.
remove-path	Deletes directories or files from the container and restores them to the host.	None	Parameter format: *hostpath***:***containerpath***[***hostpath***:***containerpath* **]** In the preceding format: *hostpath*: path on the host for storing a volume. *containerpath*: path on the container for storing a volume.
list-path	Lists all path directories in a container.	Supported options are as follows: **--pretty**: outputs data in JSON format.	None

Constraints

- When running the **add-path** command, specify an absolute path as the mount path.
- The mount point /.sharedpath is generated on the host after the mount path is specified by running the **add-path** command.
- A maximum of 128 volumes can be added to a container.
- Do not overwrite the root directory (/) in a container with the host directory by running the **add-path** command. Otherwise, the function is affected.

Example

- Start a system container, and set **hook spec** to the isulad hook execution script.

```
[root@localhost ~]# isula run -tid --hook-spec /etc/isulad-tools/hookspec.json
--system-container --external-rootfs /root/root-fs none init
e45970a522d1ea0e9cfe382c2b868d92e7b6a55be1dd239947dda1ee55f3c7f7
```

- Use isulad-tools to mount a directory on the host to a container, implementing resource sharing.

```
[root@localhost ~]# isulad-tools add-path e45970a522d1
/home/test123:/home/test123
Add path (/home/test123) to container(e45970a522d1,/home/test123) done.
```

- Create a file in the **/home/test123** directory on the host and check whether the file can be accessed in the container.

```
[root@localhost ~]# echo "hello world" > /home/test123/helloworld
[root@localhost ~]# isula exec e45970a522d1 bash
[root@localhost /]# cat /home/test123/helloworld
hello world
```

- Use isulad-tools to delete the mount directory from the container.

```
[root@localhost ~]# isulad-tools remove-path e45970a522d1
/home/test123:/home/test123
Remove path (/home/test123) from container(e45970a522d1,/home/test123) done
[root@localhost ~]# isula exec e45970a522d1 bash
[root@localhost /]# ls /home/test123/helloworld
ls: cannot access '/home/test123/helloworld': No such file or directory
```

2.4 Appendix

2.4.1 Command Line Interface List

This section lists commands in system containers, which are different from those in common containers. For details about other commands, refer to sections related to the iSulad container engine or run the **isula** *XXX* **--help** command.

Command	Parameters	Value Description
isula create/run	--external-rootfs	• Variable of the string type. • Absolute path on the host. • Specifies the rootfs of a VM when running a system container.
	--system-container	• Boolean variable. • Specifies whether a container is a system container. In a system container scenario, this function must be enabled.
	--add-host	• Variable of the string type. • Specifies the hosts configuration for a container. The format is *hostname*:*ip*. Multiple values can be set.
	--dns, --dns-option,	• Variable of the string type.

Command	Parameters	Value Description
	--dns-search	• Specifies the DNS configuration for a container. Multiple values can be set.
	--ns-change-opt	• Variable of the string type. • Container namespace kernel parameter. The value can only be **net** or **ipc**. If multiple values are set, separate them with commas (,), for example, **--ns-change-opt=net,ipc**.
	--oom-kill-disable	• Boolean variable. • Indicates whether to enable the oom-kill-disable function.
	--shm-size	• Variable of the string type. • Sets the size of **/dev/shm**. The default value is **64 MB**. The unit can be byte (B), kilobyte (KB), megabyte (MB), gigabyte (GB), terabyte (TB), or petabyte (PB).
	--sysctl	• Variable of the string type. • Specifies container kernel parameters. The format is **key=value**. Multiple values can be set. The sysctl whitelist is as follows: **kernel.msgmax**, **kernel.msgmnb**, **kernel.msgmni**, **kernel.sem**, **kernel.shmall**, **kernel.shmmax**, **kernel.shmmni**, **kernel.shm_rmid_forced**, **kernel.pid_max**, **net.**, and **fs.mqueue** **NOTE** The **kernel.pid_max** kernel parameter in a container must be able to be namespaced. Otherwise, an error is reported. Parameter restrictions (including the parameter types and value ranges) of the sysctl whitelist in a container must be the same as those of kernel parameters in the physical machine.
	--env-target-file	• Variable of the string type. • Specifies the **env** persistent file path. (The path must be an absolute path and the file must be in the rootfs directory.) The file size cannot exceed 10 MB. If the value of **--env** conflicts with that of **env** in the file, the value of **--env** takes effect. • The root directory of the absolute path is the rootfs root directory. That is, to set the file path to **/etc/environment** in the container, you need to specify **env-target-file=/etc/environment** only.
	--cgroup-parent	• Variable of the string type. • Specifies the cgroup parent directory of a container. The cgroup root directory is **/sys/fs/cgroup/***controller*.
	--host-channel	• Variable of the string type. • Specifies the memory space shared between the host

Command	Parameters	Value Description
		and a container (tmpfs). The format is as follows: *host path*:*container path*:*rw/ro*:*size limit*
	--files-limit	• Variable of the string type. • Specifies the maximum number of file handles in a container. The value must be an integer.
	--user-remap	• Variable of the string type. • The parameter format is *uid*:*gid*:*offset*.

3 Secure Container

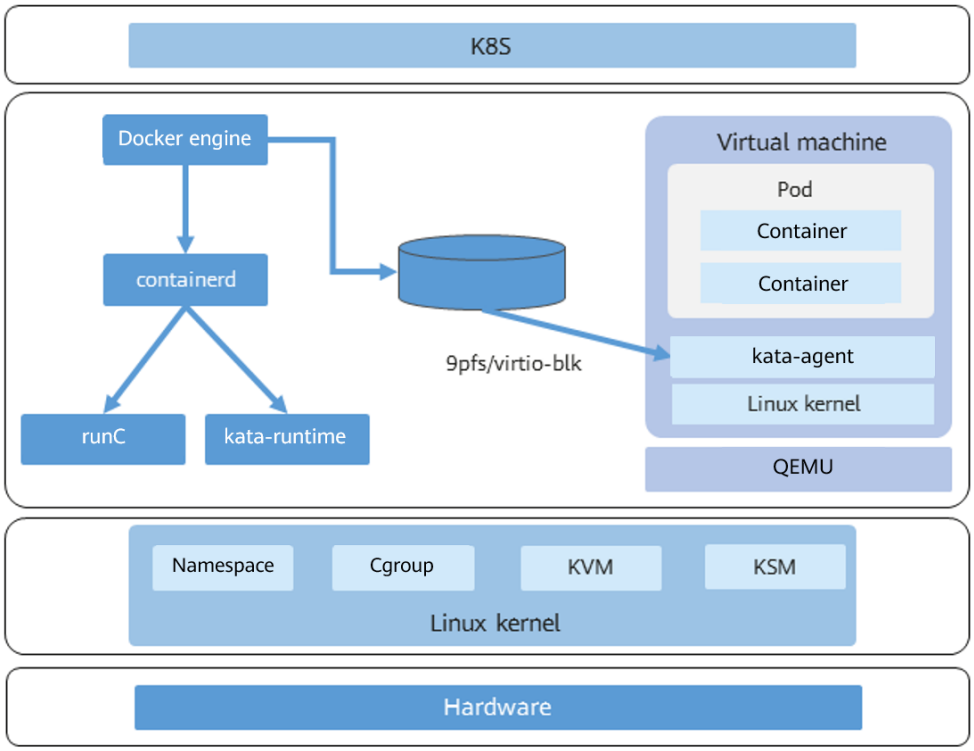
3.1 Overview

The secure container technology is an organic combination of virtualization and container technologies. Compared with a common Linux container, a secure container has better isolation performance.

Common Linux containers use namespaces to isolate the running environment between processes and use cgroups to limit resources. Essentially, these common Linux containers share the same kernel. Therefore, if a single container affects the kernel intentionally or unintentionally, the containers on the same host will be affected.

Secure containers are isolated by the virtualization layers. Containers on the same host do not affect each other.

Figure 3-1 Secure container architecture



Secure containers are closely related to the concept of pod in Kubernetes. Kubernetes is the open-source ecosystem standard for the container scheduling management platform. It defines a group of container runtime interfaces (CRIs).

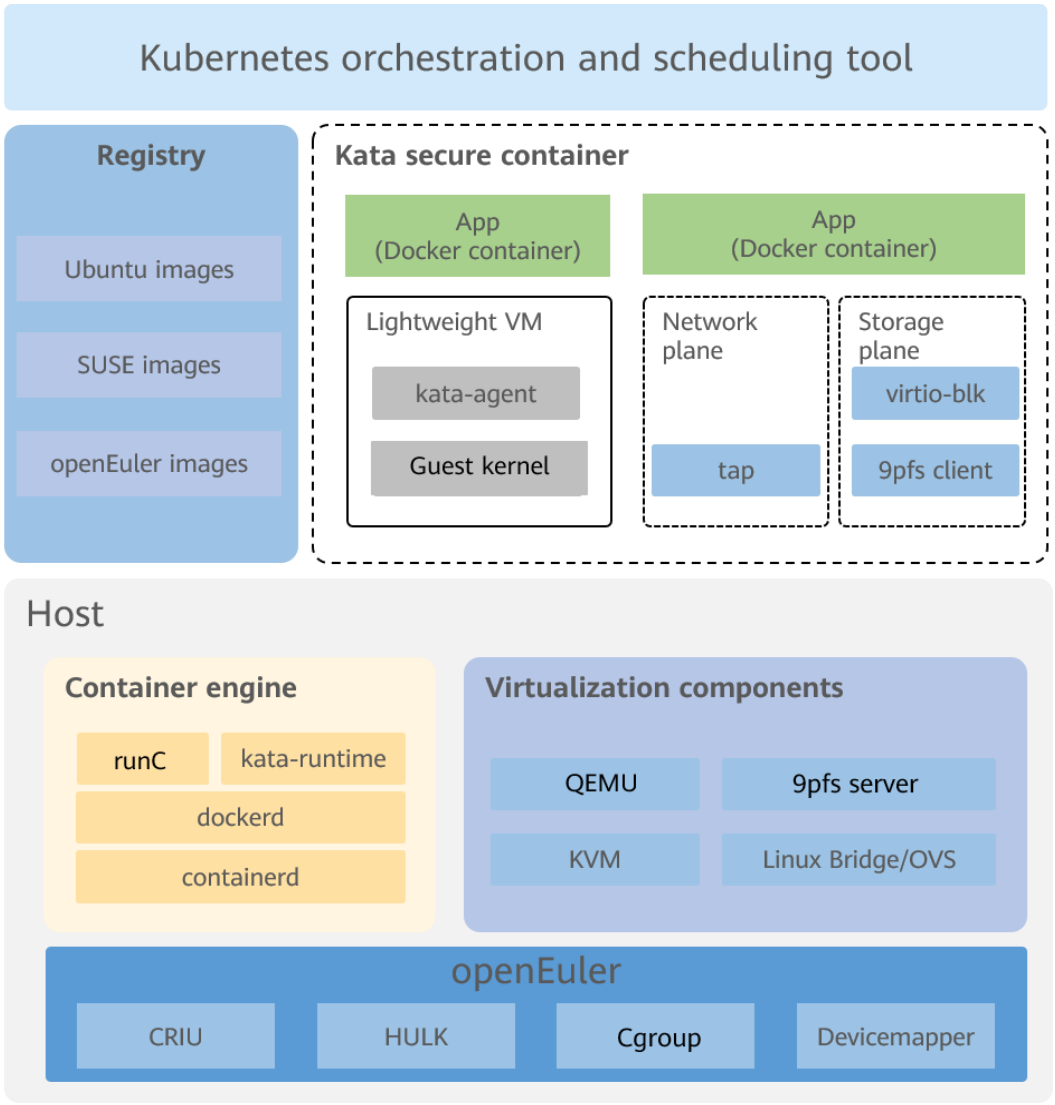
In the CRI standards, a pod is a logical grouping of one or more containers, which are scheduled together and share interprocess communication (IPC) and network namespaces. As the smallest unit for scheduling, a pod must contain a pause container and one or more service containers. The lifecycle of a pause container is the same as that of the pod.

A lightweight virtual machine (VM) in a secure container is a pod. The first container started in the VM is the pause container, and the containers started later are service containers.

In a secure container, you can start a single container or start a pod.

Figure 3-2 shows the relationship between the secure container and peripheral components.

Figure 3-2 Relationship between the secure container and peripheral components



3.2 Installation and Deployment

3.2.1 Installation Methods

Prerequisites

- For better performance experience, a secure container needs to run on the bare metal server and must not run on VMs.
- A secure container depends on the following components (openEuler 1.0 version). Ensure that the required components have been installed in the environment. To install iSulad, refer to 1.2.1 Installation Methods.
  - docker-engine
  - qemu

Installation Procedure

Released secure container components are integrated in the **kata-containers-*version*.rpm** package. You can run the **rpm** command to install the corresponding software.

```
rpm -ivh kata-containers-<version>.rpm
```

3.2.2 Deployment Configuration

3.2.2.1 Configuring the Docker Engine

To enable the Docker engine to support kata-runtime, perform the following steps to configure the Docker engine:

1. Ensure that all software packages (**docker-engine** and **kata-containers**) have been installed in the environment.
2. Stop the Docker engine.

   ```
   systemctl stop docker
   ```

3. Modify the configuration file **/etc/docker/daemon.json** of the Docker engine and add the following configuration:

   ```
   {
     "runtimes": {
       "kata-runtime": {
         "path": "/usr/bin/kata-runtime",
         "runtimeArgs": [
            "--kata-config",
            "/usr/share/defaults/kata-containers/configuration.toml"
          ]
       }
     }
   }
   ```

4. Restart the Docker engine.

   ```
   systemctl start docker
   ```

3.2.2.2 iSulad Configuration

To enable the iSulad to support the new container runtime kata-runtime, perform the following steps which are similar to those for the container engine docker-engine:

1. Ensure that all software packages (iSulad and kata-containers) have been installed in the environment.
2. Stop iSulad.

   ```
   systemctl stop isulad
   ```

3. Modify the **/etc/isulad/daemon.json** configuration file of the iSulad and add the following configurations:

   ```
   {
     "runtimes": {
       "kata-runtime": {
         "path": "/usr/bin/kata-runtime",
         "runtime-args": [
            "--kata-config",
            "/usr/share/defaults/kata-containers/configuration.toml"
          ]
   ```

```
    }
  }
}
```

4. Restart iSulad.

```
systemctl start isulad
```

3.2.2.3 Configuration.toml

The secure container provides a global configuration file configuration.toml. Users can also customize the path and configuration options of the secure container configuration file.

In the **runtimeArges** field of Docker engine, you can use **--kata-config** to specify a private file. The default configuration file path is **/usr/share/defaults/kata-containers/configuration.toml**.

The following lists the common fields in the configuration file. For details about the configuration file options, see 3.4.1 configuration.toml.

1. hypervisor.qemu
   - **path**: specifies the execution path of the virtualization QEMU.
   - **kernel**: specifies the execution path of the guest kernel.
   - **initrd**: specifies the guest initrd execution path.
   - **machine_type**: specifies the type of the analog chip. The value is **virt** for the ARM architecture and **pc** for the x86 architecture.
   - **kernel_params**: specifies the running parameters of the guest kernel.
2. proxy.kata
   - **path**: specifies the kata-proxy running path.
   - **enable_debug**: enables the debugging function for the kata-proxy process.
3. agent.kata
   - **enable_blk_mount**: enables guest mounting of the block device.
   - **enable_debug**: enables the debugging function for the kata-agent process.
4. runtime
   - **enable_cpu_memory_hotplug**: enables CPU and memory hot swap.
   - **enable_debug**: enables debugging for the kata-runtime process.

3.3 Application Scenarios

This section describes how to use a secure container.

3.3.1 Managing the Lifecycle of a Secure Container

3.3.1.1 Starting a Secure Container

You can use the Docker engine or iSulad as the container engine of the secure container. The invoking methods of the two engines are similar. You can select either of them to start a secure container.

To start a secure container, perform the following steps:

1. Ensure that the secure container component has been correctly installed and deployed.
2. Prepare the container image. If the container image is busybox, run the following commands to download the container image using the Docker engine or iSulad:

   ```
   docker pull busybox
   isula pull busybox
   ```

3. Start a secure container. Run the following commands to start a secure container using the Docker engine and iSulad:

   ```
   docker run -tid --runtime kata-runtime --network none busybox <command>
   isula run -tid --runtime kata-runtime --network none busybox <command>
   ```

   NOTE

   The secure container supports the CNI network only and does not support the CNM network. The **-p** and **--expose** options cannot be used to expose container ports. When using a secure container, you need to specify the **--net=none** option.

4. Start a pod.
   a. Start the pause container and obtain the sandbox ID of the pod based on the command output. Run the following commands to start a pause container using the Docker engine and iSulad:

      ```
      docker run -tid --runtime kata-runtime --network none --annotation
      io.kubernetes.docker.type=podsandbox <pause-image> <command>
      isula run -tid --runtime kata-runtime --network none --annotation
      io.kubernetes.cri.container-type=sandbox <pause-image> <command>
      ```

   b. Create a service container and add it to the pod. Run the following commands to create a service container using the Docker engine and iSulad:

      ```
      docker run -tid --runtime kata-runtime --network none --annotation
      io.kubernetes.docker.type=container --annotation
      io.kubernetes.sandbox.id=<sandbox-id> busybox <command>
      isula run -tid --runtime kata-runtime --network none --annotation
      io.kubernetes.cri.container-type=container --annotation
      io.kubernetes.cri.sandbox-id=<sandbox-id> busybox <command>
      ```

      --annotation is used to mark the container type, which is provided by the Docker engine and iSulad, but not provided by the open-source Docker engine in the upstream community.

3.3.1.2 Stopping a Secure Container

- Run the following command to stop a secure container:

  ```
  docker stop <containenr-id>
  ```

- Stop a pod.

  When stopping a pod, note that the lifecycle of the pause container is the same as that of the pod. Therefore, stop service containers before the pause container.

3.3.1.3 Deleting a Secure Container

Ensure that the container has been stopped.

```
docker rm <container-id>
```

To forcibly delete a running container, run the **-f** command.

```
docker rm -f <container-id>
```

3.3.1.4 Running a New Command in the Container

The pause container functions only as a placeholder container. Therefore, if you start a pod, run a new command in the service container. The pause container does not execute the corresponding command. If only one container is started, run the following command directly:

```
docker exec -ti <container-id> <command>
```

NOTE

1. If the preceding command has no response because another host runs the **docker restart** or **docker stop** command to access the same container, you can press **Ctrl**+**P**+**Q** to exit the operation.
2. If the **-d** option is used, the command is executed in the background and no error information is displayed. The exit code cannot be used to determine whether the command is executed correctly.

3.3.2 Configuring Resources for a Secure Container

The secure container runs on a virtualized and isolated lightweight VM. Therefore, resource configuration is divided into two parts: resource configuration for the lightweight VM, that is, host resource configuration; resource configuration for containers in the VM, that is, guest container resource configuration. The following describes resource configuration for the two parts in detail.

3.3.2.1 Sharing Resources

Because the secure container runs on a virtualized and isolated lightweight VM, resources in some namespaces on the host cannot be accessed. Therefore, **--net host**, **--ipc host**, **--pid host**, and **--uts host** are not supported during startup.

When a pod is started, all containers in the pod share the same net namespace and ipc namespace by default. If containers in the same pod need to share the pid namespace, you can use Kubernetes to configure the pid namespace. In Kubernetes 1.11, the pid namespace is disabled by default.

3.3.2.2 Limiting CPU Resources

1. Configure CPU resources for running a lightweight VM.

   Configuring CPU resources of a lightweight VM is to configure the vCPUs for running the VM. The secure container uses **--annotation com.github.containers.virtcontainers.sandbox_cpu** to configure the CPU resources for running the lightweight VM. This option can be configured only on the pause container.

   ```
   docker run -tid --runtime kata-runtime --network none --annotation
   io.kubernetes.docker.type=podsandbox --annotation
   com.github.containers.virtcontainers.sandbox cpu=<cpu-nums> <pause-image>
   <command>
   ```

   Example:

   ```
   #Start a pause container.
   docker run -tid --runtime kata-runtime --network none --annotation
   io.kubernetes.docker.type=podsandbox --annotation
   com.github.containers.virtcontainers.sandbox cpu=4 busybox sleep 999999
   be3255a3f66a35508efe419bc52eccd3b000032b9d8c9c62df611d5bdc115954

   #Access the container and check whether the number of CPUs is the same as that
   configured in the com.github.containers.virtcontainers.sandbox_cpu file.
   ```

```
docker exec be32 lscpu
Architecture:       aarch64
Byte Order:         Little Endian
CPU(s):            4
On-line CPU(s) list: 0-3
Thread(s) per core:  1
Core(s) per socket:  1
Socket(s):          4
```

NOTE

The maximum number of CPUs that can be configured is the number of CPUs (excluding isolated cores) that can run on the OS. The minimum number of CPUs is 0.5.

2. Configure CPU resources for running a container.

   The method of configuring CPU resources for a container is the same as that for an open-source Docker container. You can configure CPU resources by setting the following parameters in the **docker run** command:

Parameter	Description
--cpu-shares	Sets the percentage of CPU time that can be used by the container.
--cpus	Sets the number of CPUs that can be used by the container.
--cpu-period	Sets the scheduling period of the container process.
--cpu-quota	Sets the CPU time that can be used by the container process in a scheduling period.
--cpuset-cpus	Sets the list of CPUs that can be used by the container process. **NOTE** When the secure container uses the **--cpuset-cpus** option to bind a CPU, the CPU ID cannot exceed the number of CPUs in the lightweight VM corresponding to the secure container minus 1. (The CPU ID in the lightweight VM starts from 0.)
--cpuset-mems	Sets the memory node that can be accessed by the container process. **NOTE** Secure containers do not support the multi-NUMA architecture and configuration. The **--cpuset-mems** option of NUMA memory can only be set to **0**.

3. Configure CPU hot swap.

NOTE

The CPU hot swap function of the secure container requires the virtualization component QEMU.

The **enable_cpu_memory_hotplug** option in the kata-runtime configuration file **config.toml** is used to enable or disable CPU and memory hot swap. The default value is

false, indicating that CPU and memory hot swap is disabled. If the value is **true**, CPU and memory hot swap is enabled.

The **--cpus** option is reused in kata-runtime to implement the CPU hot swap function. The total number of **--cpus** options of all containers in a pod is calculated to determine the number of CPUs to be hot added to the lightweight VM.

Example:

```
#Start a pause container. By default, one vCPU is allocated to a lightweight VM.
docker run -tid --runtime kata-runtime --network none --annotation
io.kubernetes.docker.type=podsandbox busybox sleep 999999
77b40fb72f63b11dd3fcab2f6dabfc7768295fced042af8c7ad9c0286b17d24f

#View the number of CPUs in the lightweight VM after the pause container is started.
docker exec 77b40fb72f6 lscpu
Architecture:        x86 64
CPU op-mode(s):      32-bit, 64-bit
Byte Order:          Little Endian
CPU(s):             1
On-line CPU(s) list:  0
Thread(s) per core:   1
Core(s) per socket:   1
Socket(s):           1

#Start a new container in the same pod and run the --cpus command to set the number
of CPUs required by the container to 4.
docker run -tid --runtime kata-runtime --network none --cpus 4 --annotation
io.kubernetes.docker.type=container --annotation
io.kubernetes.sandbox.id=77b40fb72f63b11dd3fcab2f6dabfc7768295fced042af8c7ad9c0
286b17d24f busybox sleep 999999
7234d666851d43cbdc41da356bf62488b89cd826361bb71d585a049b6cedafd3

#View the number of CPUs in the current lightweight VM.
docker exec 7234d6668 lscpu
Architecture:        x86 64
CPU op-mode(s):      32-bit, 64-bit
Byte Order:          Little Endian
CPU(s):             4
On-line CPU(s) list:  0-3
Thread(s) per core:   1
Core(s) per socket:   1
Socket(s):           4

#View the number of CPUs in the lightweight VM after deleting the container where
CPUs are hot added.
docker rm -f 7234d666851d
7234d666851d

docker exec 77b40fb72f6  lscpu
Architecture:        x86 64
CPU op-mode(s):      32-bit, 64-bit
Byte Order:          Little Endian
CPU(s):             1
On-line CPU(s) list:  0
Thread(s) per core:   1
```

```
Core(s) per socket:   1
Socket(s):            1
```

NOTE

The pause container is only a placeholder container and does not have any workload. Therefore, when a lightweight VM is started, the CPU allocated by default can be shared by other containers. Therefore, you only need to hot add three CPUs to the lightweight VM for the new container started in the preceding example.

- After the container where the CPU is hot added is stopped, the CPU is removed when the container is started.

3.3.2.3 Limiting Memory Resources

1. Configure memory resources for running a lightweight VM.

   Configuring the memory resources of a lightweight VM is to configure the memory for running the VM. The secure container uses **--annotation com.github.containers.virtcontainers.sandbox_mem** to configure the memory resources for running the lightweight VM. This option can be configured only on the pause container.

   ```
   docker run -tid --runtime kata-runtime --network none --annotation
   io.kubernetes.docker.type=podsandbox --annotation
   com.github.containers.virtcontainers.sandbox mem=<memory-size> <pause-image>
   <command>
   ```

   Example:

   ```
   #Start a pause container and use --annotation
   com.github.containers.virtcontainers.sandbox_mem=4G to allocate 4 GB memory to the
   lightweight VM.
   docker run -tid --runtime kata-runtime --network none --annotation
   io.kubernetes.docker.type=podsandbox --annotation
   com.github.containers.virtcontainers.sandbox mem=4G busybox sleep 999999
   1532c3e59e7a45cd6b419aa1db07dd0069b0cdd93097f8944177a25e457e4297

   #View the memory information of the lightweight VM and check whether the memory size
   is the same as that configured in the
   com.github.containers.virtcontainers.sandbox_mem file.
   docker exec 1532c3e free -m
              total        used        free      shared  buff/cache   available
   Mem:          3950          20        3874          41          55        3858
   Swap:            0           0           0
   ```

   NOTE

   - If the memory size of a lightweight VM is not set using **--annotation com.github.containers.virtcontainers.sandbox_mem**, the lightweight VM uses 1 GB memory by default.
   - The minimum memory size of a pod in a secure container is 1 GB, and the maximum memory size is 256 GB. If the memory size allocated to a user exceeds 256 GB, an undefined error may occur. Currently, secure containers do not support the scenario where the memory size exceeds 256 GB.

2. Configure memory resources for running a container.

   The method of configuring memory resources for running a container is the same as that for the open-source Docker container. You can configure memory resource limitation parameters in the **docker run** command.

Parameter	Description
-m/--memory	Sets the memory size that can be used by the container process. **NOTE** • When memory hot swap is disabled, the value of **-m** must be less than or equal to the memory size allocated when the lightweight VM is started.

3. Configure memory hot add.

   The memory hot add function is also configured by the **enable_cpu_memory_hotplug** option in the kata-runtime configuration file **config.toml**. For details, see 3.

   NOTE

   Currently, memory resources support hot add only.

   The **-m** option is reused in kata-runtime to implement the memory hot add function. The sum of the **-m** options of all containers in a pod is collected to determine the number of memories to be hot added to a lightweight VM.

   Example:

```
#Start a pause container. By default, 1 GB memory is allocated to the lightweight
VM.
docker run -tid --runtime kata-runtime --network none --annotation
io.kubernetes.docker.type=podsandbox busybox sleep 999999
99b78508ada3fa7dcbac457bb0f6e3784e64e7f7131809344c5496957931119f

#View the memory size of the lightweight VM after the pause container is started.
docker exec 99b78508ada free -m
              total        used        free      shared  buff/cache   available
Mem:            983          18         914          36          50         908
Swap:             0           0           0

#Start a new container in the same pod and run the -m command to set the memory size
required by the container to 4 GB.
docker run -tid --runtime kata-runtime --network none -m 4G --annotation
io.kubernetes.docker.type=container --annotation
io.kubernetes.sandbox.id=99b78508ada3fa7dcbac457bb0f6e3784e64e7f7131809344c5496
957931119f busybox sleep 999999
c49461745a712b2ef3127fdf43b2cbb034b7614e6060b13db12b7a5ff3c830c8

#View the memory size of the lightweight VM.
docker exec c49461745 free -m
              total        used        free      shared  buff/cache   available
Mem:           4055          69        3928          36          57        3891
Swap:             0           0           0

#After deleting the container where the CPU is hot added, check the memory size of
the lightweight VM.
docker rm -f c49461745
c49461745

#The hot added memory does not support the hot add function. Therefore, after the
```

```
hot added memory container is deleted from the lightweight VM, the memory is still
4 GB.
docker exec 99b78508ada free -m
              total        used        free      shared  buff/cache   available
Mem:           4055          69        3934          36          52        3894
Swap:             0           0           0
```

NOTE

The pause container is only a placeholder container and does not have any workload. Therefore, the memory allocated to the lightweight VM during startup can be shared by other containers. You only need to hot add 3 GB memory to the lightweight VM for the new container started in the preceding example.

3.3.2.4 Limiting Block I/O Resources

1. Configure the block I/O resources for running a lightweight VM.

   To configure block I/O resources for running a lightweight VM of secure containers, use **--annotation com.github.containers.virtcontainers.blkio_cgroup**. This option can be configured only on the pause container.

   ```
   docker run -tid --runtime --network none --annotation
   io.kubernetes.docker.type=podsandbox --annotation
   com.github.containers.virtcontainers.blkio cgroup=<blkio json
   string<pause-image> <command>
   ```

   The value of **--annotation com.github.containers.virtcontainers.blkio_cgroup** must comply with the definition of the BlkioCgroup structure.

   ```
   // BlkioCgroup for Linux cgroup 'blkio' data exchange
   type BlkioCgroup struct {
       // Items specifies per cgroup values
       Items []BlockIOCgroupItem `json:"blkiocgroup,omitempty"`
   }

   type BlockIOCgroupItem struct {
       // Path represent path of blkio device
       Path string `json:"path,omitempty"`
       // Limits specifies the blkio type and value
       Limits []IOLimit `json:"limits,omitempty"`
   }

   type IOLimit struct {
       // Type specifies IO type
       Type string `json:"type,omitempty"`
       // Value specifies rate or weight value
       Value uint64 `json:"value,omitempty"`
   }
   ```

   The values of the **Type** field in the **IOLimit** structure body are as follows:

   ```
   // BlkioThrottleReadBps is the key to fetch throttle read bps
   BlkioThrottleReadBps = "throttle read bps"

   // BlkioThrottleWriteBps is the key to fetch throttle write bps
   BlkioThrottleWriteBps = "throttle write bps"

   // BlkioThrottleReadIOPS is the key to fetch throttle read iops
   BlkioThrottleReadIOPS = "throttle read iops"
   ```

```
// BlkioThrottleWriteIOPS is the key to fetch throttle write iops
BlkioThrottleWriteIOPS = "throttle write iops"

// BlkioWeight is the key to fetch blkio weight
BlkioWeight = "blkio weight"

// BlkioLeafWeight is the key to fetch blkio leaf weight
BlkioLeafWeight = "blkio_leaf_weight"
```

Example:

```
docker run -tid --runtime kata-runtime --network none --annotation
com.github.containers.virtcontainers.blkio cgroup='{"blkiocgroup":[{"path":"/de
v/sda","limits":[{"type":"throttle read bps","value":400},{"type":"throttle wri
te bps","value":400},{"type":"throttle read iops","value":700},{"type":"throttl
e write iops","value":699}]},{"limits":[{"type":"blkio weight","value":78}]}]}'
busybox sleep 999999
```

The preceding command is used to limit the block I/O traffic of the **/dev/sda** disk used by the started secure container by setting **throttle_read_bps** to 400 bit/s, **throttle_write_bps** to 400 bit/s, **throttle_read_iops** to 700 times/s, **throttle_write_iops** to 699 times/s, and the weight of the block I/O cgroup to 78.

3.3.2.5 Limiting File Descriptor Resources

To prevent the file descriptor resources on the host from being exhausted when a large number of files in the 9p shared directory are opened in the container, the secure container can customize the maximum number of file descriptors that can be opened by the QEMU process of the secure container.

The secure container reuses the **--files-limit** option in the **docker run** command to set the maximum number of file descriptors that can be opened by the QEMU process of the secure container. This parameter can be configured only on the pause container. The usage method is as follows:

```
docker run -tid --runtime kata-runtime --network none --annotation
io.kubernetes.docker.type=podsandbox --files-limit <max-open-files> <pause-image>
bash
```

NOTE

- If the value of **--files-limit** is less than the default minimum value **1024** and is not **0**, the maximum number of file descriptors that can be opened by the QEMU process of the secure container is set to the minimum value **1024**.
- If the value of **--files-limit** is 0, the maximum number of file descriptors that can be opened by the QEMU process of the secure container is the default value obtained by dividing the maximum number of file descriptors that can be opened by the system (/proc/sys/fs/file-max) by 400.
- If the maximum number of file descriptors that can be opened by the QEMU process of the secure container is not displayed when the secure container is started, the maximum number of file descriptors that can be opened by the QEMU process of the secure container is the same as the system default value.

3.3.3 Configuring Networking for a Secure Container

TAP-based Network Support

The secure container technology is implemented based on QEMU VMs. For a physical machine system, a secure container is equivalent to a VM. Therefore, the secure container may connect the VM to an external network in the Neutron network by using the test access

point (TAP) technology. You do not need to pay attention to TAP device creation and bridging. You only need to hot add the specified TAP device (with an existing host) to the VM in the pause container and update the NIC information.

Related commands are as follows:

1. **Run the following command to add a TAP NIC for a started container:**

```
$ cat ./test-iface.json | kata-runtime kata-network add-iface 6ec7a98 -
```

In the preceding command, **6ec7a98** is the truncated container ID, and **test-infs.json** is the file that describes the NIC information. The following is an example:

```
{
    "device": "tap-test",
    "name": "eth-test",
    "IPAddresses": [
        {
            "address": "172.16.0.3",
            "mask": "16"
        }
    ],
    "hwAddr":"02:42:20:6f:a3:69",
    "mtu": 1500,
    "vhostUserSocket":"/usr/local/var/run/openvswitch/vhost-user1",
    "queues":5
}
```

The fields in the JSON file are described as follows:

Field	Mandatory/Optional	Description
device	Mandatory	Name of the NIC on a host. The value can contain a maximum of 15 characters, including letters, digits, underscores (_), hyphens (-), and periods (.). It must start with a letter. The device name must be unique on the same host.
name	Mandatory	Name of the NIC in the container. The value can contain a maximum of 15 characters, including letters, digits, underscores (_), hyphens (-), and periods (.). It must start with a letter. The name must be unique in the same sandbox.
IPAddresses	Optional	IP address of the NIC. Currently, one IP address can be configured for each NIC. If no IP address is configured for the NIC, no IP address will be configured in the container, either.
hwAddr	Mandatory	MAC address of the NIC.
mtu	Mandatory	MTU of the NIC. The value ranges from 46 to 9600.
vhostUserSocket	Optional	Socket path for DPDK polling. The path contains a maximum of 128 bytes. The naming rule can contain digits, letters, and hyphens (-). The path name must start with a letter.
queues	Optional	Number of NIC queues. If this parameter is not set,

Field	Mandatory/Optional	Description
		the default value **0** is used.

The following describes the output of the **kata-runtime kata-network add-iface** command for adding NICs:

- If the command is successfully executed, the NIC information in JSON format is returned from **standard output (stdout)**. The content in JSON format is the same as the input NIC information.

  Example:

  ```
  $ kata-runtime kata-network add-iface <container-id> net.json
  {"device":"tap test","name":"eth-test","IPAddresses":[{"Family":2,"Address
  ":"173.85.100.1","Mask":"24"}],"mtu":1500,"hwAddr":"02:42:20:6e:03:01","pc
  iAddr":"01.0/00"}
  ```

- If the command fails to be executed, null is returned from **stdout**.

  Example:

  ```
  $ kata-runtime kata-network add-iface <container-id> netbad.json 2>/dev/null
  null
  ```

NOTE

If an IP address is specified for an NIC that is successfully added, Kata adds a default route whose destination is in the same network segment as the IP address of the NIC. In the preceding example, after the NIC is added, the following route is added to the container:

```
[root@6ec7a98 /]# ip route
172.16.0.0/16 dev eth-test proto kernel scope link src 172.16.0.3
```

2. **Run the following command to view the added NICs:**

   ```
   $ kata-runtime kata-network list-ifaces 6ec7a98
   [{"name":"eth-test","mac":"02:42:20:6f:a3:69","ip":["172.16.0.3/16"],"mtu":1500
   }]
   ```

   The information about the added NICs is displayed.

   The following describes the output of the **kata-runtime kata-network list-ifaces** command for listing added NICs:

   - If the command is executed successfully, information about all NICs inserted into the pod in JSON format is returned from **stdout**.

     If multiple NICs are inserted into the pod, the NIC information in JSON array format is returned.

     ```
     $ kata-runtime kata-network list-ifaces <container-id>
     [{"name":"container eth","mac":"02:42:20:6e:a2:59","ip":["172.17.25.23/8"]
     ,"mtu":1500},{"name":"container eth 2","mac":"02:90:50:6b:a2:29","ip":["19
     2.168.0.34/24"],"mtu":1500}]
     ```

     If no NIC is inserted into the pod, null is returned from **stdout**.

     ```
     $ kata-runtime kata-network list-ifaces <container-id>
     null
     ```

   - If the command fails to be executed, null is returned from **stdout**, and error description is returned from **standard error (stderr)**.

     Example:

```
$ kata-runtime kata-network list-ifaces <container-id>
null
```

3. **Add a route for a specified NIC.**

```
$ cat ./test-route.json | kata-runtime kata-network add-route 6ec7a98 -
[{"dest":"default","gateway":"172.16.0.1","device":"eth-test"}]
```

The following describes the output of the **kata-runtime kata-network add-route** command for adding a route to a specified NIC:

- If the command is executed successfully, the added route information in JSON format is returned from **stdout**.

  Example:

```
$ kata-runtime kata-network add-route <container-id> route.json
[{"dest":"177.17.0.0/24","gateway":"177.17.25.1","device":"netport test 1"
}]
```

- If the command fails to be executed, null is returned from **stdout**, and error description is returned from **standard error (stderr)**.

  Example:

```
$ kata-runtime kata-network add-route <container-id> routebad.json 2>/dev/null
null
```

Key fields are described as follows:

- **dest**: Network segment corresponding to the route. The value is in the format of <*ip*>/<*mask*>. <*ip*> is mandatory. There are three cases:
  i. Both IP address and mask are configured.
  ii. If only an IP address is configured, the default mask is 32.
  iii. If **"dest":"default"** is configured, there is no destination by default. In this case, the gateway needs to be configured.
- **gateway**: Next-hop gateway of the route. When **"dest":"default"** is configured, the gateway is mandatory. In other cases, this parameter is optional.
- **device**: Name of the NIC corresponding to the route, which is mandatory. The value contains a maximum of 15 characters.

NOTE

If a route is added for the loopback device **lo** in the container, the device name corresponding to the **device** field in the route configuration file is **lo**.

4. **Run the following command to delete a specified route:**

```
$ cat ./test-route.json | kata-runtime kata-network del-route 6ec7a98 -
```

The fields in the **test-route.json** file are the same as those in the JSON file for adding a route.

The following describes the output of the **kata-runtime kata-network del-route** command for deleting a specified route:

- If the command is executed successfully, the added route information in JSON format is returned from **stdout**.

  Example:

```
$ kata-runtime kata-network del-route <container-id> route.json
[{"dest":"177.17.0.0/24","gateway":"177.17.25.1","device":"netport test 1"
}]
```

- If the command fails to be executed, null is returned from **stdout**, and error description is returned from **standard error (stderr)**.

Example:

```
$ kata-runtime kata-network del-route <container-id> routebad.json 2>/dev/null
null
```

NOTE

- In the input fields, **dest** is mandatory, and both **device** and **gateway** are optional. Kata performs fuzzy match based on different fields and deletes the corresponding routing rules. For example, if **dest** is set to an IP address, all rules of this IP address will be deleted.
- If the route of the loopback device **lo** in the container is deleted, the device name corresponding to the **device** field in the route configuration file is **lo**.

5. **Run the following command to delete an NIC:**

```
$ cat ./test-iface.json | kata-runtime kata-network del-iface 6ec7a98 -
```

NOTE

When deleting an NIC, you can only delete it based on the **name** field in the NIC container. Kata does not identify other fields.

The following describes the output of the **kata-runtime kata-network del-iface** command for deleting NICs:

- If the command is executed successfully, null is returned from **stdout**.

  Example:

  ```
  $ kata-runtime kata-network del-iface <container-id> net.json
  null
  ```

- If the command fails to be executed, the information about NICs that fail to be deleted in JSON format is returned from **stdout**, and error description is returned from **stderr**.

  Example:

  ```
  $ kata-runtime kata-network del-iface <container-id> net.json
  {"device":"tapname fun 012","name":"netport test 1","IPAddresses":[{"Famil
  y":0,"Address":"177.17.0.1","Mask":"8"}],"mtu":1500,"hwAddr":"02:42:20:6e:
  a2:59","linkType":"tap"}
  ```

The preceding are common commands. For details about the command line interfaces, see 3.4.2 APIs.

Kata IPVS Subsystem

The secure container provides an API for adding the **ipvs** command and setting the IPVS rule for the container. The functions include adding, editing, and deleting virtual services, adding, editing, and deleting real servers, querying IPVS service information, setting connection timeout, clearing the system connection cache, and importing rules in batches.

1. Add a virtual service address for the container.

```
kata-runtime kata-ipvs ipvsadm --parameters "--add-service --tcp-service
172.17.0.7:80 --scheduler rr --persistent 3000" <container-id>
```

2. Modify virtual service parameters of a container.

```
kata-runtime kata-ipvs ipvsadm --parameters "--edit-service --tcp-service
172.17.0.7:80 --scheduler rr --persistent 5000" <container-id>
```

3. Delete the virtual service address of a container.

```
kata-runtime kata-ipvs ipvsadm --parameters "--delete-service --tcp-service
172.17.0.7:80" <container-id>
```

4. Add a real server for the virtual service address.

```
kata-runtime kata-ipvs ipvsadm --parameters "--add-server --tcp-service
172.17.0.7:80 --real-server 172.17.0.4:80 --weight 100" <container-id>
```

5. Modify real server parameters of a container.

```
kata-runtime kata-ipvs ipvsadm --parameters "--edit-server --tcp-service
172.17.0.7:80 --real-server 172.17.0.4:80 --weight 200" <container-id>
```

6. Delete a real server from a container.

```
kata-runtime kata-ipvs ipvsadm --parameters "--delete-server --tcp-service
172.17.0.7:80 --real-server 172.17.0.4:80" <container-id>
```

7. Query service information.

```
kata-runtime kata-ipvs ipvsadm --parameters "--list" <container-id>
```

8. It takes a long time to import rules one by one. You can write rules into a file and import them in batches.

```
kata-runtime kata-ipvs ipvsadm --restore - < <rule file path> <container-id>
```

NOTE

By default, the NAT mode is used for adding a single real server. To add real servers in batches, you need to manually add the **-m** option to use the NAT mode.

The following is an example of the rule file content:

-A -t 10.10.11.12:100 -s rr -p 3000

-a -t 10.10.11.12:100 -r 172.16.0.1:80 -m

-a -t 10.10.11.12:100 -r 172.16.0.1:81 -m

-a -t 10.10.11.12:100 -r 172.16.0.1:82 -m

9. Clear the system connection cache.

```
kata-runtime kata-ipvs cleanup --parameters "--orig-dst 172.17.0.4 --protonum tcp"
<container-id>
```

10. Set timeout interval for TCP, TCP FIN, or UDP connections.

```
kata-runtime kata-ipvs ipvsadm --parameters "--set 100 100 200" <container-id>
```

NOTE

1. Each container supports a maximum of 20000 iptables rules (5000 services and three servers/services). Both add-service and add-server are rules.
2. Before importing rules in batches, you need to clear existing rules.
3. No concurrent test scenario exists.
4. The preceding are common commands. For details about the command line interfaces, see 3.4.2 APIs.

3.3.4 Monitoring Secure Containers

Description

The **kata events** command is used to view the status information of a specified container. The information includes but is not limited to the container memory, CPU, PID, Blkio, hugepage memory, and network information.

Usage

```
kata-runtime events [command options] <container-id>
```

Parameters

- **-- interval value**: specifies the query period. If this parameter is not specified, the default query period is 5 seconds.
- **--stats**: displays container information and exits the query.

Prerequisites

The container to be queried must be in the **running** state. Otherwise, the following error message will be displayed: "Container ID (<container_id>) does not exist".

This command can be used to query the status of only one container.

Example

- The container status is displayed every three seconds.

```
$ kata-runtime events --interval 3s 5779b2366f47
{
   "data": {
      "blkio": {},
      "cpu": {
         "throttling": {},
         "usage": {
            "kernel": 130000000,
            "percpu": [
               214098440
            ],
            "total": 214098440,
            "user": 10000000
         }
      },
      "hugetlb": {},
      "intel rdt": {},
      "interfaces": [
         {
            "name": "lo",
            "rx bytes": 0,
            "rx dropped": 0,
            "rx errors": 0,
            "rx packets": 0,
            "tx bytes": 0,
            "tx dropped": 0,
            "tx errors": 0,
            "tx packets": 0
         }
      ],
      "memory": {
         "cache": 827392,
         "kernel": {
            "failcnt": 0,
            "limit": 9223372036854771712,
            "max": 421888,
            "usage": 221184
         },
         "kernelTCP": {
```

```
            "failcnt": 0,
            "limit": 0
        },
        "raw": {
            "active anon": 49152,
            "active file": 40960,
            "cache": 827392,
            "dirty": 0,
            "hierarchical memory limit": 9223372036854771712,
            "hierarchical memsw limit": 9223372036854771712,
            "inactive anon": 0,
            "inactive file": 839680,
            "mapped file": 540672,
            "pgfault": 6765,
            "pgmajfault": 0,
            "pgpgin": 12012,
            "pgpgout": 11803,
            "rss": 4096,
            "rss huge": 0,
            "shmem": 32768,
            "swap": 0,
            "total active anon": 49152,
            "total active file": 40960,
            "total cache": 827392,
            "total dirty": 0,
            "total inactive anon": 0,
            "total inactive file": 839680,
            "total mapped file": 540672,
            "total pgfault": 6765,
            "total pgmajfault": 0,
            "total pgpgin": 12012,
            "total pgpgout": 11803,
            "total rss": 4096,
            "total rss huge": 0,
            "total shmem": 32768,
            "total swap": 0,
            "total unevictable": 0,
            "total writeback": 0,
            "unevictable": 0,
            "writeback": 0
        },
        "swap": {
            "failcnt": 0,
            "limit": 9223372036854771712,
            "max": 34201600,
            "usage": 1204224
        },
        "usage": {
            "failcnt": 0,
            "limit": 9223372036854771712,
            "max": 34201600,
            "usage": 1204224
        }
    },
    "pids": {
```

```
            "current": 1
        },
        "tcp": {},
        "tcp6": {},
        "udp": {},
        "udp6": {}
    },
    "id": "5779b2366f47cd1468ebb1ba7c52cbdde3c7d3a5f2af3eefadc8356700fc860b",
    "type": "stats"
}
```

- The query exits after the container status is displayed.

```
kata-runtime events --stats <container_id>
```

The format of the command output is the same as that of the previous command. However, the output of this command is displayed only once.

3.4 Appendix

3.4.1 configuration.toml

NOTE

The value of each field in the **configuration.toml** file is subject to the **configuration.toml** file in the **kata-containers-<*version*>.rpm package**. You cannot set any field in the configuration file.

```
[hypervisor.qemu]
path: specifies the execution path of the virtualization QEMU.
kernel: specifies the execution path of the guest kernel.
initrd: specifies the guest initrd execution path.
image: specifies the execution path of the guest image (not applicable).
machine_type: specifies the type of the analog chip. The value is virt for the ARM architecture and pc for the x86 architecture.
kernel_params: specifies the running parameters of the guest kernel.
firmware: specifies the firmware path. If this parameter is left blank, the default firmware is used.
machine_accelerators: specifies an accelerator.
default_vcpus: specifies the default number of vCPUs for each SB/VM.
default_maxvcpus: specifies the default maximum number of vCPUs for each SB/VM.
default_root_ports: specifies the default number of root ports for each SB/VM.
default_bridges: specifies the default number of bridges for each SB/VM.
default_memory: specifies the default memory size of each SB/VM. The default value is 1024 MiB.
memory_slots: specifies the number of memory slots for each SB/VM. The default value is 10.
memory_offset: specifies the memory offset. The default value is 0.
disable_block_device_use: disables the block device from being used by the rootfs of the container.
shared_fs: specifies the type of the shared file system. The default value is virtio-9p.
virtio_fs_daemon: specifies the path of the vhost-user-fs daemon process.
virtio_fs_cache_size: specifies the default size of the DAX cache.
virtio_fs_cache: specifies the cache mode.
block_device_driver: specifies the driver of a block device.
block_device_cache_set: specifies whether to set cache-related options for a block device. The default value is false.
```

block_device_cache_direct: specifies whether to enable O_DIRECT. The default value is false.
block_device_cache_noflush: specifies whether to ignore device update requests. The default value is **false**.
enable_iothreads: enables iothreads.
enable_mem_prealloc: enables VM RAM pre-allocation. The default value is **false**.
enable_hugepages: enables huge pages. The default value is **false**.
enable_swap: enables the swap function. The default value is **false**.
enable_debug: enables QEMU debugging. The default value is **false**.
disable_nesting_checks: disables nested check.
msize_9p = 8192: specifies the number of bytes transmitted in each 9p packet.
use_vsock: uses vsocks to directly communicate with the agent (the prerequisite is that vsocks is supported). The default value is **false**.
hotplug_vfio_on_root_bus: enables the hot swap of the VFIO device on the root bus. The default value is **false**.
disable_vhost_net: disables vhost_net. The default value is **false**.
entropy_source: specifies the default entropy source.
guest_hook_path: specifies the binary path of the guest hook.

[factory]
enable_template: enables the VM template. The default value is **false**.
template_path: specifies the template path.
vm_cache_number: specifies the number of VM caches. The default value is 0.
vm_cache_endpoint: specifies the address of the Unix socket used by the VMCache. The default value is **/var/run/kata-containers/cache.sock**.

[proxy.kata]
path: specifies the kata-proxy running path.
enable_debug: enables proxy debugging. The default value is **false**.

[shim.kata]
path: specifies the running path of kata-shim.
enable_debug: enables shim debugging. The default value is **false**.
enable_tracing: enables shim opentracing.

[agent.kata]
enable_debug: enables the agent debugging function. The default value is **false**.
enable_tracing: enables the agent tracing function.
trace_mode: specifies the trace mode.
trace_type: specifies the trace type.
enable_blk_mount: enables guest mounting of the block device.

[netmon]
enable_netmon: enables network monitoring. The default value is **false**.
path: specifies the kata-netmon running path.
enable_debug: enables netmon debugging. The default value is **false**.

[runtime]
enable_debug: enables runtime debugging. The default value is **false**.
enable_cpu_memory_hotplug: enables CPU and memory hot swap. The default value is **false**.
internetworking_model: specifies the network interconnection mode between VMs and containers.
disable_guest_seccomp: disables the seccemp security mechanism in the guest application. The default value is **true**.
enable_tracing: enables runtime opentracing. The default value is **false**.

```
disable_new_netns: disables network namespace creation for the shim and hypervisor processes. The default value is false.
experimental: enables the experimental feature, which does not support user-defined configurations.
```

3.4.2 APIs

Table 3-1 Commands related to the kata-runtime network

Command	Subcommand	File Example	Field	Description	Remarks

kata-network
NOTE
• The kata-network command must be used in groups. Network devices that are not added using **kata-runtime kata-network** cannot be deleted or listed using **kata-runtime kata-network**. The reverse is also true.
• **kata-runtime kata-network** imports configuration parameters through a file or stdin.
	add-iface
	NOTE
	• An interface can be added to only one container.
	• The execution result is subject to the returned value (non-zero return value).
		{
		"device":"tap1",
		"name":"eth1",
		"IPAddresses":[{"address":"172.17.1.10","mask":"24"}],
		"mtu":1300,
		"hwAddr":"02:42:20:6f:a2:80"
		"vhostUserSocket":"/usr/local/var/run/openvswitch/vhost-user1"
		}
			device	Sets the name of the NIC on a host.	Mandatory. The value can contain a maximum of 15 characters, including letters, digits, underscores (_), hyphens (-), and periods (.). It must start with a letter. The device name must be unique on the same host.

			name	Sets the name of the NIC in the container.	Mandatory. The value can contain a maximum of 15 characters, including letters, digits, underscores (_), hyphens (-), and periods (.). It must start with a letter. Ensure that the name is unique in the same sandbox.
			IPAddresses	Sets the IP address of an NIC.	Optional. Currently, one IP address can be configured for each NIC. If no IP address is configured for the NIC, no IP address will be configured in the container, either.
			mtu	Sets the MTU of an NIC.	Mandatory. The value ranges from 46 to 9600.

Command	Subcommand	File Example	Field	Description	Remarks
			hwAddr	Sets the MAC address of an NIC.	Mandatory.
			vhostUserSocket	Sets the DPDK polling socket path.	Optional. The path contains a maximum of 128 bytes. The naming rule can contain digits, letters, and hyphens (-). The path name must start with a letter.

	del-iface	{
		"name":"eth1"
		}
			None	Deletes an NIC from a container.	**NOTE**
					When deleting a NIC, you can only delete it based on the name field in the NIC container. Kata does not identify other fields.

	list-ifaces	None	None	Queries the NIC list in a container.	None

	add-route	{
		"dest":"172.17.10.10/24",
		"gateway":"",
		"device":"eth1"
		}
			dest	Sets the network segment corresponding to the route.	The value is in the format of <*ip*>/<*mask*>. <*ip*> is mandatory.
					There are three cases:
					1. Both IP address and mask are configured.
					2. If only an IP address is configured, the default mask is 32.
					3. If **"dest":"default"** is configured, there is no destination by default. In this case, the gateway

Command	Subcommand	File Example	Field	Description	Remarks
					needs to be configured.
			gateway	Sets the next-hop gateway of the route.	When **"dest":"default"** is configured, the gateway is mandatory. In other cases, this parameter is optional.
			device	Sets the name of the NIC corresponding to the route.	Mandatory. The value contains a maximum of 15 characters.

	del-route	{
		"dest":"172.17.10.10/24"
		}
			None	Deletes a container routing rule.	**dest** is mandatory, and both **device** and **gateway** are optional.
					NOTE
					Kata performs fuzzy match based on different fields and deletes the corresponding routing rules.

	list-routes	None	None	Queries the route list in a container.	None

Table 3-2 kata-ipvs command line interfaces

Command	Subcommand	Field	Parameter	Sub-parameter	Description	Remarks
kata-ipvs	**ipvsadm**	--parameters	-A, --add-service	-t, --tcp-service -u, --udp-service	Virtual service type.	Mandatory. You can select **--tcp-service** or **--udp-service**. The format is **ip:port**. The value of **port** ranges from 1 to 65535. Example: `kata-runtime kata-ipvs ipvsadm --parameters "--add-service --tcp-service 172.17.0.7:80`

Command	Subcommand	Field	Parameter	Sub-parameter	Description	Remarks
						`--scheduler rr --persistent 3000" <container-id>`
				-s, --scheduler	Load balancing scheduling algorithm.	Mandatory. Value range: rr|wrr|lc|wlc|lblc|lblcr|dh|sh|sed|nq.
				-p, --persistent	Service duration.	Mandatory. The value ranges from 1 to 2678400, in seconds.
			-E, --edit-service	-t, --tcp-service -u, --udp-service	Virtual service type.	Mandatory. You can select **--tcp-service** or **--udp-service**. The format is **ip:port**. The value of **port** ranges from 1 to 65535.
				-s, --scheduler	Load balancing scheduling algorithm.	Mandatory. Value range: rr|wrr|lc|wlc|lblc|lblcr|dh|sh|sed|nq.
				-p, --persistent	Service duration.	Mandatory. The value ranges from 1 to 2678400, in seconds.
			-D, --delete-service	-t, --tcp-service -u, --udp-service	Virtual service type.	Mandatory. You can select **--tcp-service** or **--udp-service**. The format is **ip:port**. The value of **port** ranges from 1 to 65535.
			-a, --add-server	-t, --tcp-service -u, --udp-service	Virtual service type.	Mandatory. You can select **--tcp-service** or **--udp-service**. The format is **ip:port**. The value of **port** ranges from 1 to 65535. Example: `kata-runtime kata-ipvs ipvsadm --parameters "--add-server --tcp-service 172.17.0.7:80 --real-server 172.17.0.4:80`

Command	Subcommand	Field	Parameter	Sub-parameter	Description	Remarks
						`--weight 100" <container-id>`
				-r, --real-server	Real server address.	Mandatory. The format is **ip:port**. The value of **port** ranges from 1 to 65535.
				-w, --weight	Weight	Optional. The value ranges from 0 to 65535.
			-e, --edit-server	-t, --tcp-service -u, --udp-service	Virtual service type.	Mandatory. You can select **--tcp-service** or **--udp-service**. The format is **ip:port**. The value of **port** ranges from 1 to 65535.
				-r, --real-server	Real server address.	Mandatory. The format is **ip:port**. The value of **port** ranges from 1 to 65535.
				-w, --weight	Weight	Optional. The value ranges from 0 to 65535.
			-d, --delete-server	-t, --tcp-service -u, --udp-service	Virtual service type.	Mandatory. You can select **--tcp-service** or **--udp-service**. The format is **ip:port**. The value of **port** ranges from 1 to 65535.
				-r, --real-server	Real server address.	Mandatory. The format is **ip:port**. The value of **port** ranges from 1 to 65535.
			-L, --list	-t, --tcp-service -u, --udp-service	Queries virtual service information.	Optional. Example: `kata-runtime kata-ipvs ipvsadm --parameters "--list --tcp-service ip:port" <container-id>`
			--set	--tcp	TCP timeout.	Mandatory. The value ranges from 0 to 1296000. Example: `kata-runtime kata-ipvs ipvsadm --parameters "--set 100 100 200" <container-id>`

Command	Subcommand	Field	Parameter	Sub-parameter	Description	Remarks
				--tcpfin	TCP FIN timeout.	Mandatory. The value ranges from 0 to 1296000.
				--udp	UDP timeout.	Mandatory. The value ranges from 0 to 1296000.
		--restore	-		Imports standard inputs in batches. Rule files can be specified.	Example: `kata-runtime kata-ipvs ipvsadm --restore - < <rule file path> <container-id>` **NOTE** By default, the NAT mode is used for adding a single real server. To add real servers in batches, you need to manually add the **-m** option to use the NAT mode. The following is an example of the rule file content: -A -t 10.10.11.12:100 -s rr -p 3000 -a -t 10.10.11.12:100 -r 172.16.0.1:80 -m -a -t 10.10.11.12:100 -r 172.16.0.1:81 -m -a -t 10.10.11.12:100 -r 172.16.0.1:82 -m
	cleanup	--parameters	-d, --orig-dst		Specifies the IP address.	Mandatory. Example: `kata-runtime kata-ipvs cleanup --parameters "--orig-dst 172.17.0.4 --protonum tcp" <container-id>`
			-p, --protonum		Protocol type.	Mandatory. The value can be **tcp** or **udp**.

4 Docker Container

4.1 Overview

Docker is an open-source Linux container engine that enables quick application packaging, deployment, and delivery. The original meaning of Docker is dork worker, whose job is to pack the goods to the containers, and move containers, and load containers. Similarly, the job of Docker in Linux is to pack applications to containers, and deploy and run applications on various platforms using containers. Docker uses Linux Container technology to turn applications into standardized, portable, and self-managed components, enabling the "build once" and "run everywhere" features of applications. Features of Docker technology include: quick application release, easy application deployment and management, and high application density.

4.2 Installation and Deployment

4.2.1 Installation Configurations and Precautions

This section describes important configurations related to the installation of the open-source container Docker.

4.2.1.1 Precautions

- The **docker-engine** RPM package cannot be installed together with the **containerd**, **runc**, or **podman** RPM package. This is because the **docker-engine** RPM package contains all components required for Docker running, including **containerd**, **runc**, and **docker** binary files. Yet the **containerd**, **runc**, and **podman** RPM packages also contain the corresponding binary files. Software package conflicts may occur due to repeated installation.

4.2.1.2 Basic Installation Configuration

4.2.1.2.1 Daemon Parameter Configuration

You can add configuration items to the **/etc/docker/daemon.json** file to customize parameters. You can run the **dockerd --help** command to view related configuration items and their usage methods. A configuration example is as follows:

```
cat /etc/docker/daemon.json
{
    "debug": true,
    "storage-driver": "overlay2",
    "storage-opts": ["overlay2.override_kernel_check=true"]
}
```

4.2.1.2.2 Daemon Running Directory Configuration

Re-configuring various running directories and files (including **--graph** and **--exec-root**) may cause directory conflicts or file attribute changes, affecting the normal use of applications.

NOTICE

Therefore, the specified directories or files should be used only by Docker to avoid file attribute changes and security issues caused by conflicts.

- Take **--graph** as an example. When **/new/path/** is used as the new root directory of the daemon, if a file exists in **/new/path/** and the directory or file name conflicts with that required by Docker (for example, **containers**, **hooks**, and **tmp**), Docker may update the original directory or file attributes, including the owner and permission.

NOTICE

From Docker 17.05, the **--graph** parameter is marked as **Deprecated** and replaced with the **--data-root** parameter.

4.2.1.2.3 Daemon Network Configuration

- After the network segment of the docker0 bridge is specified by using the **--bip** parameter on Docker daemon, if the **--bip** parameter is deleted during the next Docker daemon restart, the docker0 bridge uses the previous value of **--bip**, even if the docker0 bridge is deleted before the restart. The reason is that Docker saves the network configuration and restores the previous configuration by default during the next restart.
- When running the **docker network create** command to concurrently create networks, you can create two networks with the same name. The reason is that Docker networks are distinguished by IDs. The name is only an alias that is easy to identify and may not be unique.
- In the Docker bridge network mode, a Docker container establishes external communication through NAT on the host. When Docker daemon starts a Docker container, a docker-proxy process is started for each port mapped on the host to access the proxy. It is recommended that you map only the necessary ports when using userland-proxy to reduce the resources consumed by the port mapping of docker-proxy.

4.2.1.2.4 Daemon umask Configuration

The default **umask** value of the main container process and exec process is **0022**. To meet security specifications and prevent containers from being attacked, the default value of **umask** is changed to **0027** after runC implementation is modified. After the modification, the other groups cannot access new files or directories.

The default value of **umask** is **0027** when Docker starts a container. You can change the value to **0022** by running the **--exec-opt native.umask=normal** command during container startup.

NOTICE

If **native.umask** is configured in **docker create** or **docker run** command, its value is used.

For details, see the parameter description in 4.6.2.4 create and 4.6.2.16 run.

4.2.1.2.5 Daemon Start Time

The Docker service is managed by systemd, which restricts the startup time of each service. If the Docker service fails to be started within the specified time, the possible causes are as follows:

- If Docker daemon is started for the first time using devicemapper, the Docker daemon needs to perform the initialization operation on the device. This operation, however, will perform a large number of disk I/O operations. When the disk performance is poor or many I/O conflicts exist, the Docker daemon startup may time out. devicemapper needs to be initialized only once and does not need to be initialized again during later Docker daemon startup.
- If the usage of the current system resources is too high, the system responses slowly, all operations in the system slow down, and the startup of the Docker service may time out.
- During the restart, a daemon traverses and reads configuration files and the init layer and writable layer configurations of each container in the Docker working directory. If there are too many containers (including the created and exited containers) in the current system and the disk read and write performance is limited, the startup of the Docker service may time out due to the long-time daemon traversing.

If the service startup times out, you are advised to rectify the fault as follows:

- Ensure that the container orchestration layer periodically deletes unnecessary containers, especially the exited containers.
- Based on performance requirements of the solution, adjust the cleanup period of the orchestration layer and the start time of the Docker service.

4.2.1.2.6 Journald Component

After systemd-journald is restarted, Docker daemon needs to be restarted. Journald obtains the Docker daemon logs through a pipe. If the journald service is restarted, the pipe is disabled. The write operation of Docker logs triggers the SIGPIPE signal, which causes the Docker daemon crash. If this signal is ignored, the subsequent Docker daemon logs may fail to be recorded. Therefore, you are advised to restart Docker daemon after the journald service is restarted or becomes abnormal, ensuring that Docker logs can be properly recorded and preventing status exceptions caused by daemon crash.

4.2.1.2.7 Firewalld Component

You need to restart the Docker service after restarting or starting firewalld.

- When the firewalld service is started, the iptables rules of the current system are cleared. Therefore, if the firewalld service is restarted during Docker daemon startup, the Docker service may fail to insert iptables rules, causing the Docker service startup failure.
- If the firewalld service is restarted after the Docker service is started, or the status of the firewalld service (service paused or resumed) is changed, the iptables rules of the Docker service are deleted. As a result, the container with port mapping fails to be created.

4.2.1.2.8 Iptables Component

If the **--icc=false** option is added in Docker, the communication between containers can be restricted. However, if the OS has some rules, the communication between containers may not be restricted. For example:

```
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
...
0    0 ACCEPT    icmp --  *     *      0.0.0.0/0          0.0.0.0/0
...
0    0 DROP      all  --  docker0 docker0  0.0.0.0/0          0.0.0.0/0
...
```

In the **Chain FORWARD** command, the ACCEPT icmp rule is added to DROP. As a result, after the **--icc=false** option is added, containers can be pinged, but the peer end is unreachable if UDP or TCP is used.

Therefore, if you want to add the **--icc=false** option when using Docker in a container OS, you are advised to clear iptables rules on the host first.

4.2.1.2.9 Audit Component

You can configure audit for Docker. However, this configuration is not mandatory. For example:

```
-w /var/lib/docker -k docker
-w /etc/docker -k docker
-w /usr/lib/systemd/system/docker.service -k docker
-w /usr/lib/systemd/system/docker.socket -k docker
-w /etc/sysconfig/docker -k docker
-w /usr/bin/docker-containerd -k docker
-w /usr/bin/docker-runc -k docker
-w /etc/docker/daemon.json -k docker
```

Configuring audit for Docker brings certain benefits for auditing, while it does not have any substantial effects on attack defense. In addition, the audit configurations cause serious efficiency problems, for example, the system may not respond smoothly. Therefore, exercise caution in the production environment.

The following uses **-w /var/lib/docker -k docker** as an example to describe how to configure Docker audit.

```
[root@localhost signal]# cat /etc/audit/rules.d/audit.rules | grep docker -w
/var/lib/docker/  -k docker
[root@localhost signal]# auditctl -R /etc/audit/rules.d/audit.rules | grep docker
[root@localhost signal]# auditctl -l | grep docker -w /var/lib/docker/ -p rwxa -k docker
```

NOTE

-p [r|w|x|a] and **-w** are used together to monitor the read, write, execution, and attribute changes (such as timestamp changes) of the directory. In this case, any file or directory operation in the **/var/lib/docker** directory will be recorded in the **audit.log** file. As a result, too many logs will be recorded in the **audit.log** file, which severely affects the memory or CPU usage of the auditd, and further affects the OS. For example, logs similar to the following will be recorded in the **/var/log/audit/audit.log** file each time the **ls /var/lib/docker/containers** command is executed:

```
type=SYSCALL msg=audit(1517656451.457:8097): arch=c000003e syscall=257 success=yes
exit=3 a0=ffffffffffffff9c a1=1b955b0 a2=90800 a3=0 items=1 ppid=17821 pid=1925 auid=0
uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts6 ses=4 comm="ls"
exe="/usr/bin/ls" subj=unconfined u:unconfined r:unconfined t:s0-s0:c0.c1023
key="docker"type=CWD msg=audit(1517656451.457:8097):  cwd="/root"type=PATH
msg=audit(1517656451.457:8097): item=0 name="/var/lib/docker/containers"
inode=1049112 dev=fd:00 mode=040700 ouid=0 ogid=0 rdev=00:00
obj=unconfined_u:object_r:container_var_lib_t:s0 objtype=NORMAL
```

4.2.1.2.10 Security Configuration seccomp

During the container network performance test, it is found that the performance of Docker is lower than that of the native kernel namespace. After seccomp is enabled, system calls (such as sendto) are not performed through system_call_fastpath. Instead, tracesys is called, which greatly deteriorates the performance. Therefore, you are advised to disable seccomp in container scenarios where services require high performance. For example:

```
docker run -itd --security-opt seccomp=unconfined busybox:latest
```

4.2.1.2.11 Do Not Modify Private Directory of Docker Daemon

Do not modify the root directory used by Docker (**/var/lib/docker** by default), the directory during operation (**/run/docker** by default), or the files or directories in the two directories. The forbidden operations include deleting files, adding files, creating soft or hard links for the directories or files, or modifying attributes, permissions, or contents of the files. If any modification is required, contact the Euler container team for review.

4.2.1.2.12 Precautions for Common Users in the Scenario Where a Large Number of Containers Are Deployed

The maximum number of processes that a common user can create on an OS host can be restricted by creating the **/etc/security/limits.d/20-nproc.conf** file in the system. Similarly, the maximum number of processes that a common user can create in a container is determined by the value in the **/etc/security/limits.d/20-nproc.conf** file in the container image, as shown in the following example:

```
cat /etc/security/limits.conf
*      soft    nproc   4096
```

If an error is reported due to insufficient resources when a large number of containers are deployed by a common user, increase the value **4096** in the **/etc/security/limits.d/20-nproc.conf** file.

Configure the maximum value based on the maximum capability of the kernel, as shown in the following example:

```
[root@localhost ~]# sysctl -a | grep pid max
kernel.pid_max = 32768
```

4.2.1.3 Storage Driver Configuration

This Docker version supports two storage drivers: overlay2 and devicemapper. Since overlay2 has better performance than devicemapper, it is recommended that overlay2 be preferentially used in the production environment.

4.2.1.3.1 overlay2 Storage Driver Configuration

Configuration Methods

overlay2 is the default storage driver of Docker. You can also use either of the following methods to check or configure the driver:

- Edit the **/etc/docker/daemon.json** file to check or configure the **storage-driver** field.

```
cat /etc/docker/daemon.json
{
   "storage-driver": "overlay2"
}
```

- Edit the **/etc/sysconfig/docker-storage** file and check or configure the Docker daemon startup parameters.

```
cat /etc/sysconfig/docker-storage
DOCKER_STORAGE_OPTIONS="--storage-driver=overlay2"
```

Precautions

- When you perform lifecycle management operations on some containers, an error may be reported, indicating that the corresponding rootfs or executable file cannot be found.
- If the health check of a container is configured to execute executable files in the container, an error may be reported, which causes the health check failure of the container.
- When you use overlay2 as the graphdriver and modify an image file in a container for the first time, the modification fails if the file size is greater than the remaining space of the system. Even if a little modification on the file is involved, the whole file must be copied to the upper layer. If the remaining space is insufficient, the modification fails.
- Compared with common file systems, the overlay2 file system has the following behavior differences:
  - Kernel version

    overlay2 is compatible only with the native kernel 4.0 or later. You are advised to use the Ext4 file system.
  - Copy-UP performance

    Modifying files at the lower layer triggers file replication to the upper layer. Data block replication and fsync are time-consuming.
  - Rename directories
    - The rename system call is allowed only when both the source and the destination paths are at the merged layer. Otherwise, the EXDEV error is reported.
    - Kernel 4.10 introduces the redirect directory feature to fix this issue. The corresponding kernel option is **CONFIG_OVERLAY_FS_REDIRECT_DIR**.

      When overlay2 is used, a file system directory fails to be renamed because the related feature configured in the

/sys/module/overlay/parameters/redirect_dir file has been disabled. To use this feature, you need to manually set **/sys/module/overlay/parameters/redirect_dir** to **Y**.

- Hard link disconnection
  - If there are multiple hard links in the lower-layer directory, writing data to the merged layer will trigger Copy-UP, resulting in hard link disconnection.
  - The index feature is introduced in kernel 4.13 to fix this issue. The corresponding kernel option is **CONFIG_OVERLAY_FS_INDEX**. Note that this option is not forward compatible and does not support hot upgrade.
- Changes of **st_dev** and **st_ino**

  After Copy-UP is triggered, you can view only new files at the merged layer, and inodes change. Although **attr** and **xattr** can be replicated, **st_dev** and **st_ino** are unique and cannot be replicated. As a result, you can run **stat** and **ls** commands to check inode changes accordingly.
- fd change

  Before Copy-UP is triggered, you can obtain the descriptor fd1 when opening a file in read-only mode. After Copy-UP is trigger, you can obtain the descriptor fd2 when opening the file with the same name. The two descriptors point to different files. The data written to fd2 is not displayed in fd1.

Abnormal Scenarios

When a container uses the overlay2 storage driver, mount points may be overwritten.

Abnormal Scenario: Mount Point Being Overwritten

In the faulty container, there is a mount point in **/var/lib/docker/overlay2**.

```
[root@localhost ~]# mount -l | grep overlay
overlay on
/var/lib/docker/overlay2/844fd3bca8e616572935808061f009d106a8748dfd29a0a4025645457
fa21785/merged type overlay
(rw,relatime,seclabel,lowerdir=/var/lib/docker/overlay2/l/JL5PZQLNDCIBU3ZOG3LPPDBH
IJ:/var/lib/docker/overlay2/l/ELRPYU4JJG4FDPRLZJCZZE4UO6,upperdir=/var/lib/docker/
overlay2/844fd3bca8e616572935808061f009d106a8748dfd29a0a4025645457fa21785/diff,wor
kdir=/var/lib/docker/overlay2/844fd3bca8e616572935808061f009d106a8748dfd29a0a40256
45457fa21785/work)
/dev/mapper/dm-root on /var/lib/docker/overlay2 type ext4
(rw,relatime,seclabel,data=ordered)
```

An error as follows may occur when some Docker commands are executed:

```
[root@localhost ~]# docker rm 1348136d32
docker rm: Error response from daemon: driver "overlay2" failed to remove root filesystem
for 1348136d32: error while removing
/var/lib/docker/overlay2/844fd3bca8e616572935808061f009d106a8748dfd29a0a4025645457
fa21785: invalid argument
```

You will find that the rootfs of the corresponding container cannot be found on the host. However, this does not mean that the rootfs is lost. The rootfs is overwritten by the mount point in **/var/lib/docker/overlay2**, and services are still running properly. The solutions are as follows:

- Solution 1
  a. Run the following command to check the graphdriver used by Docker:

     ```
     docker info | grep "Storage Driver"
     ```

  b. Run the following commands to query the current mount point:

     ```
     Devicemapper: mount -l | grep devicemapper
     Overlay2: mount -l | grep overlay2
     ```

     The output format is *A* on *B* type *C* (*D*).
     - *A*: block device name or **overlay**
     - *B*: mount point
     - *C*: file system type
     - *D*: mounting attribute

  c. Run the **umount** command on the mount points (*B*) one by one from bottom to top.
  d. Run the **docker restart** command on all the containers or delete all the containers.
  e. Run the following command to restart Docker:

     ```
     systemctl restart docker
     ```

- Solution 2
  a. Migrate services.
  b. Restart nodes.

4.2.1.3.2 devicemapper Storage Driver Configuration

If you need to set the storage driver of Docker to devicemapper, you can also use either of the following methods to check or configure the driver:

- Edit the **/etc/docker/daemon.json** file to check or configure the **storage-driver** field.

  ```
  cat /etc/docker/daemon.json
  {
      "storage-driver": "devicemapper"
  }
  ```

- Edit the **/etc/sysconfig/docker-storage** file and check or configure the Docker daemon startup parameters.

  ```
  cat /etc/sysconfig/docker-storage
  DOCKER_STORAGE_OPTIONS="--storage-driver=devicemapper"
  ```

Precautions

- To use devicemapper, you must use the direct-lvm mode. For details about the configuration method, refer to https://docs.docker.com/engine/userguide/storagedriver/device-mapper-driver/#configure-direct-lvm-mode-for-production.
- When configuring devicemapper, if the system does not have sufficient space for automatic capacity expansion of thinpool, disable the automatic capacity expansion function.
- Do not set both the following two parameters in the **/etc/lvm/profile/docker-thinpool.profile** file to **100**:

  ```
  activation {
    thin_pool_autoextend_threshold=80
  ```

```
  thin pool autoextend percent=20
}
```

- You are advised to add **--storage-opt dm.use_deferred_deletion=true** and **--storage-opt dm.use_deferred_removal=true** when using devicemapper.
- When devicemapper is used, you are advised to use Ext4 as the container file system. You need to add **--storage-opt dm.fs=ext4** to the configuration parameters of Docker daemon.
- If graphdriver is devicemapper and the metadata files are damaged and cannot be restored, you need to manually restore the metadata files. Do not directly operate or tamper with metadata of the devicemapper storage driver in Docker daemon.
- When the devicemapper LVM is used, if the devicemapper thinpool is damaged due to abnormal power-off, you cannot ensure the data integrity or whether the damaged thinpool can be restored. Therefore, you need to rebuild the thinpool.

Precautions for Switching the devicemapper Storage Pool When the User Namespace Feature Is Enabled on Docker Daemon

- Generally, the path of the deviceset-metadata file is **/var/lib/docker/devicemapper/metadata/deviceset-metadata** during container startup.
- If user namespaces are used, the path of the deviceset-metadata file is **/var/lib/docker/***userNSUID.GID***/devicemapper/metadata/deviceset-metadata**.
- When you use the devicemapper storage driver and the container is switched between the user namespace scenario and common scenario, the **BaseDeviceUUID** content in the corresponding deviceset-metadata file needs to be cleared. In the thinpool capacity expansion or rebuild scenario, you also need to clear the **BaseDeviceUUID** content in the deviceset-metadata file. Otherwise, the Docker service fails to be restarted.

4.2.1.4 Impact of Forcibly Killing Docker Background Processes

The call chain of Docker is long. Forcibly killing docker background processes (such as sending **kill -9**) may cause data status inconsistency. This section describes some problems that may be caused by forcible killing.

4.2.1.4.1 Semaphores May Be Residual

When the devicemapper is used as the graphdriver, forcible killing may cause residual semaphores. Docker creates semaphores when performing operations on devicemapper. If daemon is forcibly killed before the semaphores are released, the release may fail. A maximum of one semaphore can be leaked at a time, and the leakage probability is low. However, the Linux OS has an upper limit on semaphores. When the number of semaphore leakage times reaches the upper limit, new semaphores cannot be created. As a result, Docker daemon fails to be started. The troubleshooting method is as follows:

1. Check the residual semaphores in the system.

```
$ ipcs
------ Message Queues --------
key       msqid     owner     perms     used-bytes   messages
------ Shared Memory Segments --------
key       shmid     owner     perms     bytes     nattch    status
------ Semaphore Arrays --------
key       semid     owner     perms     nsems
0x0d4d3358 238977024  root      600       1
0x0d4d0ec9 270172161  root      600       1
0x0d4dc02e 281640962  root      600       1
```

2. Run the **dmsetup** command to check semaphores created by devicemapper. The semaphore set is the subset of the system semaphores queried in the previous step.

```
$ dmsetup udevcookies
```

3. Check the upper limit of kernel semaphores. The fourth value is the upper limit of the current system semaphores.

```
$ cat /proc/sys/kernel/sem
250     32000   32      128
```

If the number of residual semaphores in step 1 is the same as the upper limit of semaphores in step 3, the number of residual semaphores reaches the upper limit. In this case, Docker daemon cannot be normally started. You can run the following command to increase the upper limit to restart Docker:

```
$ echo 250 32000  32  1024 > /proc/sys/kernel/sem
```

You can also run the following command to manually clear the residual devicemapper semaphores. The following describes how to clear the devicemapper semaphores applied one minute ago.

```
$ dmsetup udevcomplete all 1
This operation will destroy all semaphores older than 1 minutes with keys that have
a prefix 3405 (0xd4d).
Do you really want to continue? [y/n]: y
0 semaphores with keys prefixed by 3405 (0xd4d) destroyed. 0 skipped.
```

4.2.1.4.2 NICs May Be Residual

When a container is started in bridge mode, forcibly killing may cause residual NICs. In bridge network mode, when Docker creates a container, a pair of veths are created on the host, and then the NIC information is saved to the database. If daemon is forcibly killed before the NIC information is saved to the database of Docker, the NIC cannot be associated with Docker and cannot be deleted during the next startup because Docker deletes unused NICs from its database.

4.2.1.4.3 Failed to Restart a Container

If container hook takes a long time, and containerd is forcibly killed during container startup, the container start operation may fail. When containerd is forcibly killed during container startup, an error is returned for the Docker start operation. After containerd is restarted, the last startup may still be in the **runc create** execution phase (executing the user-defined hook may take a long time). If you run the **docker start** command again to start the container, the following error message may be displayed:

```
Error response from daemon: oci runtime error: container with id exists: xxxxxx
```

This error is caused by running **runc create** on an existing container (or being created). After the **runc create** operation corresponding to the first start operation is complete, the **docker start** command can be successfully executed.

The execution of hook is not controlled by Docker. In this case, if the container is recycled, the containerd process may be suspended when an unknown hook program is executed. In addition, the risk is controllable (although the creation of the current container is affected in a short period).

- After the first operation is complete, the container can be successfully started again.
- Generally, a new container is created after the container fails to be started. The container that fails to be started cannot be reused.

In conclusion, this problem has a constraint on scenarios.

4.2.1.4.4 Failed to Restart the Docker Service

The Docker service cannot be restarted properly due to frequent startup in a short period The Docker system service is monitored by systemd. If the Docker service is restarted for more than five times within 10s, the systemd service detects the abnormal startup. Therefore, the Docker service is disabled. Docker can respond to the restart command and be normally restarted only when the next period of 10s starts.

4.2.1.5 Impact of System Power-off

When a system is unexpectedly powered off or system panic occurs, Docker daemon status may not be updated to the disk in time. As a result, Docker daemon is abnormal after the system is restarted. The possible problems include but are not limited to the following:

- A container is created before the power-off. After the restart, the container is not displayed when the **docker ps –a** command is run, as the file status of the container is not updated to the disk. As a result, daemon cannot obtain the container status after the restart.
- Before the system power-off, a file is being written. After daemon is restarted, the file format is incorrect or the file content is incomplete. As a result, loading fails.
- As Docker database (DB) will be damaged during power-off, all DB files in **data-root** will be deleted during node restart. Therefore, the following information created before the restart will be deleted after the restart:
  - Network: Resources created through Docker network will be deleted after the node is restarted.
  - Volume: Resources created through Docker volume will be deleted after the node is restarted.
  - Cache construction: The cache construction information will be deleted after the node is restarted.
  - Metadata stored in containerd: Metadata stored in containerd will be recreated when a container is started. Therefore, the metadata stored in containerd will be deleted when the node is restarted.

NOTE

If you want to manually clear data and restore the environment, you can set the environment variable **DISABLE_CRASH_FILES_DELETE** to **true** to disable the function of clearing DB files when the daemon process is restarted due to power-off.

4.3 Container Management

4.3.1 Creating a Container

Downloading Images

Only user **root** can run the **docker** command. If you log in as a common user, you need to use the **sudo** command before running the **docker** command.

```
[root@localhost ~]# docker pull busybox
```

This command is used to download the **busybox:latest** image from the official Docker registry. (If no tag is specified in the command, the default tag name **latest** is used.) During the image download, the system checks whether the dependent layer exists locally. If yes, the image download is skipped. When downloading images from a private registry, specify the registry description. For example, if a private registry containing some common images is created and its IP address is **192.168.1.110:5000**, you can run the following command to download the image from the private registry:

```
[root@localhost ~]# docker pull 192.168.1.110:5000/busybox
```

The name of the image downloaded from the private registry contains the registry address information, which may be too long. Run the **docker tag** command to generate an image with a shorter name.

```
[root@localhost ~]# docker tag 192.168.1.110:5000/busybox busybox
```

Run the **docker images** command to view the local image list.

Running a Simple Application

```
[root@localhost ~]# docker run busybox /bin/echo "Hello world"
Hello world
```

This command uses the **busybox:latest** image to create a container, and executes the **echo "Hello world"** command in the container. Run the following command to view the created container:

```
[root@localhost ~]# docker ps -l
CONTAINER ID       IMAGE             COMMAND                CREATED           STATUS
PORTS             NAMES
d8c0a3315bc0       busybox"/bin/echo 'Hello wo..."   5 seconds ago      Exited (0) 3
seconds ago                    practical_franklin
```

Creating an Interactive Container

```
[root@localhost ~]# docker run -it busybox /bin/bash
root@bf22919af2cf:/# ls
bin  boot  dev  etc  home  lib  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp
usr  var
root@bf22919af2cf:/# pwd
/
```

The **-ti** option allocates a pseudo terminal to the container and uses standard input (STDIN) for interaction. You can run commands in the container. In this case, the container is an independent Linux VM. Run the **exit** command to exit the container.

Running a Container in the Background

Run the following command. **-d** indicates that the container is running in the background. **--name=container1** indicates that the container name is **container1**.

```
[root@localhost ~]# docker run -d --name=container1 busybox /bin/sh -c "while true;do
echo hello world;sleep 1;done"
7804d3e16d69b41aac5f9bf20d5f263e2da081b1de50044105b1e3f536b6db1c
```

The command output contains the container ID but does not contain **hello world**. In this case, the container is running in the background. You can run the **docker ps** command to view the running container.

```
[root@localhost ~]# docker ps
CONTAINER ID       IMAGE             COMMAND               CREATED           STATUS
PORTS             NAMES
7804d3e16d69       busybox "/bin/sh -c 'while tr"   11 seconds ago     Up 10 seconds
container1
```

Run the following **docker logs** command to view the output during container running:

```
[root@localhost ~]# docker logs container1
hello world
hello world
hello world
...
```

Container Network Connection

By default, a container can access an external network, while port mapping is required when an external network accesses a container. The following uses how to run the private registry service in Docker as an example. In the following command, **-P** is used to expose open ports in the registry to the host.

```
[root@localhost ~]# docker run --name=container registry -d -P registry
cb883f6216c2b08a8c439b3957fb396c847a99079448ca741cc90724de4e4731
```

The container_registry container has been started, but the mapping between services in the container and ports on the host is not clear. You need to run the **docker port** command to view the port mapping.

```
[root@localhost ~]# docker port container registry
5000/tcp -> 0.0.0.0:49155
```

The command output shows that port 5000 in the container is mapped to port 49155 on the host. You can access the registry service by using the host IP address **49155**. Enter **http://localhost:49155** in the address box of the browser and press **Enter**. The registry version information is displayed.

When running registry images, you can directly specify the port mapping, as shown in the following:

```
docker run --name=container_registry -d -p 5000:5000 registry
```

-p 5000:5000 is used to map port 5000 in the container to port 5000 on the host.

Precautions

- **Do Not Add -a stdin Independently During Container Startup**

  When starting a container, you must add **-a stdout** or **-a stderr** together with **-a stdin** instead of **-a stdin** only. Otherwise, the device stops responding even after the container exits.

- **Do Not Use the Long Or Short ID of an Existing Container As the Name of a New Container**

  When creating a container, do not use the long or short ID of the existing container A as the name of the new container B. If the long ID of container A is used as the name of container B, Docker will match container A even though the name of container B is used as the specified target container for operations. If the short ID of container A is used as the name of container B, Docker will match container B even though the short ID of container A is used as the specified target container for operations. This is because

Docker matches the long IDs of all containers first. If the matching fails, the system performs exact matching using the value of **container_name**. If matching failure persists, the container ID is directly matched in fuzzy mode.

- **Containers That Depend on Standard Input and Output, Such As sh/bash, Must Use the -ti Parameter to Avoid Exceptions**

  Normal case: If you do not use the **-ti** parameter to start a process container such as sh/bash, the container exits immediately.

  The cause of this problem is that Docker creates a stdin that matches services in the container first. If the interactive parameters such as **-ti** are not set, Docker closes pipe after the container is started and the service container process sh/bash exits after stdin is closed.

  Exception: If Docker daemon is forcibly killed in a specific phase (before pipe is closed), daemon of the pipe is not closed in time. In this case, the sh/bash process does not exit even without **-ti**. As a result, an exception occurs. You need to manually clear the container.

  After being restarted, daemon takes over the original container stream. Containers without the **-ti** parameter may not be able to process the stream because these containers do not have streams to be taken over in normal cases. In actual services, sh/bash without the **-ti** parameter does not take effect and is seldom used. To avoid this problem, the **-ti** parameter is used to restrict interactive containers.

- **Container Storage Volumes**

  If you use the **-v** parameter to mount files on the host to a container when the container is started, the inodes of the files may be changed when you run the **vi** or **sed** command to modify the files on the host or in the container. As a result, files on the host and in the container are not synchronized. Do not mount files in the container in this mode (or do not use together with the **vi** and **sed** commands). You can also mount the upper-layer directories of the files to avoid exceptions. The **nocopy** option can be used to prevent original files in the mount point directory of a container from being copied to the source directory of the host when Docker mounts volumes. However, this option can be used only when an anonymous volume is mounted and cannot be used in the bind mount scenario.

- **Do Not Use Options That May Affect the Host**

  The **--privileged** option enables all permissions for a container. On the container, mounting operations can be performed and directories such as **/proc** and **/sys** can be modified, which may affect the host. Therefore, do not use this option for common containers.

  A host-shared namespace, such as the **--pid host**, **--ipc host**, or **--net host** option, can enable a container to share the namespace with the host, which will also affect the host. Therefore, do not use this option.

- **Do Not Use the Unstable Kernel Memory Cgroup**

  Kernel memory cgroup on the Linux kernel earlier than 4.0 is still in the experimental phase and runs unstably. Therefore, do not use kernel memory cgroup.

  When the **docker run --kernel-memory** command is executed, the following alarm is generated:

  ```
  WARNING: You specified a kernel memory limit on a kernel older than 4.0. Kernel memory limits are experimental on older kernels, it won't work as expected as expected and can cause your system to be unstable.
  ```

- **blkio-weight Parameter Is Unavailable in the Kernel That Supports blkio Precise Control**

--blkio-weight-device can implement more accurate blkio control in a container. The control requires a specified disk device, which can be implemented through the **--blkio-weight-device** parameter of Docker. In this kernel, Docker does not provide the **--blkio-weight** mode to limit the container blkio. If you use this parameter to create a container, the following error is reported:

```
docker: Error response from daemon: oci runtime error: container linux.go:247:
starting container process caused "process linux.go:398: container init caused
\"process linux.go:369: setting cgroup config for ready process caused
\\\"blkio.weight not supported, use weight_device instead\\\"\""
```

- **Using --blkio-weight-device in CFQ Scheduling Policy**

  The **--blkio-weight-device** parameter works only when the disk works in the Completely Fair Queuing (CFQ) policy.

  You can view the scheduler file (**/sys/block/***disk***/queue/scheduler**) to obtain the policies supported by the disk and the current policy. For example, you can run the following command to view **sda**.

  ```
  # cat /sys/block/sda/queue/scheduler noop [deadline] cfq
  ```

  sda supports the following scheduling policies: **noop**, **deadline**, and **cfq**, and the **deadline** policy is being used. You can run the **echo** command to change the policy to **cfq**.

  ```
  # echo cfq > /sys/block/sda/queue/scheduler
  ```

- **systemd Usage Restrictions in Basic Container Images**

  When containers created from basic images are used, systemd in basic images is used only for system containers.

Concurrent Performance

- There is an upper limit for the message buffer in Docker. If the number of messages exceeds the upper limit, the messages are discarded. Therefore, it is recommended that the number of commands executed concurrently should not exceed 1000. Otherwise, the internal messages in Docker may be lost and the container may fail to be started.
- When containers are concurrently created and restarted, the error message"oci runtime error: container init still running" is occasionally reported. This is because containerd optimizes the performance of the event waiting queue. When a container is stopped, the **runc delete** command is executed to kill the init processes in the container within 1s. If the init processes are not killed within 1s, runC returns this error message. The garbage collection (GC) mechanism of containerd reclaims residual resources after **runc delete** is executed at an interval of 10s. Therefore, operations on the container are not affected. If the preceding error occurs, wait for 4 or 5s and restart the container.

Security Feature Interpretation

1. The following describes default permission configuration analysis of Docker.

   In the default configuration of a native Docker, capabilities carried by each default process are as follows:

   ```
   "CAP CHOWN",
   "CAP DAC OVERRIDE",
   "CAP FSETID",
   "CAP FOWNER",
   "CAP MKNOD",
   "CAP NET RAW",
   "CAP SETGID",
   "CAP_SETUID",
   ```

```
"CAP SETFCAP",
"CAP SETPCAP",
"CAP NET BIND SERVICE",
"CAP SYS CHROOT",
"CAP KILL",
"CAP_AUDIT_WRITE",
```

The default seccomp configuration is a whitelist. If any syscall is not in the whitelist, **SCMP_ACT_ERRNO** is returned by default. Different system invoking is enabled for different caps of Docker. If a capability is not in the whitelist, Docker will not assign it to the container by default.

2. CAP_SYS_MODULE

   CAP_SYS_MODULE allows a container to insert the ko module. Adding this capability allows the container to escape or even damage the kernel. Namespace provides the maximum isolation for a container. In the ko module, you only need to point its namespace to **init_nsproxy**.

3. CAP_SYS_ADMIN

   The sys_admin permission provides the following capabilities for a container:

   - For file system: **mount**, **umount**, and **quotactl**
   - For namespace setting: **setns**, **unshare**, and **clone new namespace**
   - driver ioctl
   - For PCI control: **pciconfig_read**, **pciconfig_write**, and **pciconfig_iobase**
   - **sethostname**

4. CAP_NET_ADMIN

   CAP_NET_ADMIN allows a container to access network interfaces and sniff network traffic. The container can obtain the network traffic of all containers including the host, which greatly damages network isolation.

5. CAP_DAC_READ_SEARCH

   CAP_DAC_READ_SEARCH calls the open_by_handle_at and name_to_handle_at system calls. If the host is not protected by SELinux, the container can perform brute-force search for the inode number of the file_handle structure to open any file on the host, which affects the isolation of the file system.

6. CAP_SYS_RAWIO

   CAP_SYS_RAWIO allows a container to write I/O ports to the host, which may cause the host kernel to crash.

7. CAP_SYS_PTRACE

   The ptrace permission for a container provides ptrace process debugging in the container. RunC has fixed this vulnerability. However, some tools, such as nsenter and docker-enter, are not protected. In a container, processes executed by these tools can be debugged to obtain resource information (such as namespace and fd) brought by these tools. In addition, ptrace can bypass seccomp, greatly increasing attack risks of the kernel.

8. Docker capability interface: --cap-add all

   --cap-add all grants all permissions to a container, including the dangerous permissions mentioned in this section, which allows the container to escape.

9. Do not disable the seccomp feature of Docker.

   Docker has a default seccomp configuration with a whitelist. **sys_call** that is not in the whitelist is disabled by seccomp. You can disable the seccomp feature by running **--security-opt 'seccomp:unconfined'**. If seccomp is disabled or the user-defined

seccomp configuration is used but the filtering list is incomplete, attack risks of the kernel in the container are increased.

10. Do not set the **/sys** and **/proc** directories to writable.

    The **/sys** and **/proc** directories contain Linux kernel maintenance parameters and device management interfaces. If the write permission is configured for the directories in a container, the container may escape.

11. Docker open capability: --CAP_AUDIT_CONTROL

    The permission allows a container to control the audit system and run the **AUDIT_TTY_GET** and **AUDIT_TTY_SET** commands to obtain the TTY execution records (including the **root** password) recorded in the audit system.

12. CAP_BLOCK_SUSPEND and CAP_WAKE_ALARM

    The permission provides a container the capability to block the system from suspending (epoll).

13. CAP_IPC_LOCK

    With this permission, a container can break the max locked memory limit in **ulimit** and use any mlock large memory block to cause DoS attacks.

14. CAP_SYS_LOG

    In a container with this permission, system kernel logs can be read by using dmesg to break through kernel kaslr protection.

15. CAP_SYS_NICE

    In a container with this permission, the scheduling policy and priority of a process can be changed, causing DoS attacks.

16. CAP_SYS_RESOURCE

    With this permission, a container can bypass resource restrictions, such as disk space resource restriction, keymaps quantity restriction, and **pipe-size-max** restriction, causing DoS attacks.

17. CAP_SYS_TIME

    In a container with this capability, the time on the host can be changed.

18. Risk analysis of Docker default capabilities

    The default capabilities of Docker include CAP_SETUID and CAP_FSETID. If the host and a container share a directory, the container can set permissions for the binary file in the shared directory. Common users on the host can use this method to elevate privileges. With the CAP_AUDIT_WRITE capability, a container can write logs to the host, and the host must be configured with log anti-explosion measures.

19. Docker and host share namespace parameters, such as **--pid**, **--ipc**, and **--uts**.

    This parameter indicates that the container and host share the namespace. The container can attack the host as the namespace of the container is not isolated from that of the host. For example, if you use **--pid** to share PID namespace with the host, the PID on the host can be viewed in the container, and processes on the host can be killed at will.

20. **--device** is used to map the sensitive directories or devices of the host to the container.

    The Docker management plane provides interfaces for mapping directories or devices on a host to the container, such as **--device** and **-v**. Do not map sensitive directories or devices on the host to the container.

4.3.2 Creating Containers Using hook-spec

Principles and Application Scenarios

Docker supports the extended features of hooks. The execution of hook applications and underlying runC complies with the OCI standards. For details about the standards, visit https://github.com/opencontainers/runtime-spec/blob/master/config.md#hooks.

There are three types of hooks: prestart, poststart, and poststop. They are respectively used before applications in the container are started, after the applications are started, and after the applications are stopped.

API Reference

The **--hook-spec** parameter is added to the **docker run** and **create** commands and is followed by the absolute path of the **spec** file. You can specify the hooks to be added during container startup. These hooks will be automatically appended after the hooks that are dynamically created by Docker (currently only libnetwork prestart hook) to execute programs specified by users during the container startup or destruction.

The structure of **spec** is defined as follows:

```
// Hook specifies a command that is run at a particular event in the lifecycle of a
container
type Hook struct{
            Path    string   `json:"path"`
            Args    []string `json:"args,omitempty"`
            Env     []string `json:"env,omitempty"`
            Timeout *int     `json:"timeout,omitempty"`
}
// Hooks for container setup and teardown
type  Hooks struct{
            // Prestart is a list of hooks to be run before the container process is
executed.
            // On Linux, they are run after the container namespaces are created.
            Prestart []Hook `json:"prestart,omitempty"`
            // Poststart is a list of hooks to be run after the container process is
started.
            Poststart []Hook `json:"poststart,omitempty"`
            // Poststop is a list of hooks to be run after the container process exits.
            Poststop []Hook `json:"poststop,omitempty"`
}
```

- The **Path**, **Args**, and **Env** parameters are mandatory.
- **Timeout** is optional, while you are advised to set this parameter to a value ranging from 1 to 120. The parameter type is int. Floating point numbers are not allowed.
- The content of the **spec** file must be in JSON format as shown in the preceding example. If the format is incorrect, an error is reported.
- Both **docker run --hook-spec /tmp/hookspec.json** *xxx*, and **docker create --hook-spec /tmp/hookspec.json** *xxx* **&& docker start** *xxx* can be used.

Customizing Hooks for a Container

Take adding a NIC during the startup as an example. The content of the **hook spec** file is as follows:

```
{
   "prestart": [
       {
          "path": "/var/lib/docker/hooks/network-hook",
          "args": ["network-hook", "tap0", "myTap"],
          "env": [],
          "timeout": 5
       }
    ],
    "poststart":[],
    "poststop":[]
}
```

Specify prestart hook to add the configuration of a network hook. The path is **/var/lib/docker/hooks/network-hook**. **args** indicates the program parameters. Generally, the first parameter is the program name, and the second parameter is the parameter accepted by the program. For the network-hook program, two parameters are required. One is the name of the NIC on the host, and the other is the name of the NIC in the container.

- Precautions
  a. The **hook** path must be in the **hooks** folder in the **graph** directory (**--graph**) of Docker. Its default value is **/var/lib/docker/hooks**. You can run the **docker info** command to view the root path.

     ```
     [root@localhost ~]# docker info
     ...
     Docker Root Dir: /var/lib/docker
     ...
     ```

     This path may change due to the user's manual configuration and the use of user namespaces (**daemon --userns-remap**). After the symbolic link of the path is parsed, the parsed path must start with *Docker Root Dir*/**hooks** (for example, **/var/lib/docker/hooks**). Otherwise, an error message is displayed.
  b. The **hook** path must be an absolute path because daemon cannot properly process a relative path. In addition, an absolute path meets security requirements.
  c. The information output by the hook program to stderr is output to the client and affects the container lifecycle (for example, the container may fail to be started). The information output to stdout is ignored.
  d. Do not reversely call Docker instructions in hooks.
  e. The execute permission must have been granted on the configured hook execution file. Otherwise, an error is reported during hook execution.
  f. The execution time of the hook operation must be as short as possible. If the prestart period is too long (more than 2 minutes), the container startup times out. If the poststop period is too long (more than 2 minutes), the container is abnormal.

     The known exceptions are as follows: When the **docker stop** command is executed to stop a container and the clearing operation is performed after 2 minutes, the hook operation is not complete. Therefore, the system waits until the hook operation is complete (the process holds a lock). As a result, all operations related to the container stop responding. The operations can be recovered only after the hook operation is complete. In addition, the two-minute timeout processing of the **docker stop** command is an asynchronous process. Therefore, even if the **docker stop** command is successfully executed, the container status is still **up**. The container status is changed to **exited** only after the hook operation is completed.

- Suggestions
  a. You are advised to set the hook timeout threshold to a value less than 5s.
  b. You are advised to configure only one prestart hook, one poststart hook, and one poststop hook for each container. If too many hooks are configured, the container startup may take a long time.
  c. You are advised to identify the dependencies between multiple hooks. If required, you need to adjust the sequence of the hook configuration files according to the dependencies. The execution sequence of hooks is based on the sequence in the configured **spec** file.

Multiple hook-spec

If multiple hook configuration files are available and you need to run multiple hooks, you must manually combine these files into a configuration file and specify the new configuration file by using the **--hook-spec** parameter. Then all hooks can take effect. If multiple **--hook-spec** parameters are configured, only the last one takes effect.

Configuration examples:

The content of the **hook1.json** file is as follows:

```
# cat /var/lib/docker/hooks/hookspec.json
{
   "prestart": [
       {
          "path": "/var/lib/docker/hooks/lxcfs-hook",
          "args": ["lxcfs-hook", "--log", "/var/log/lxcfs-hook.log"],
          "env": []
       }
    ],
    "poststart":[],
    "poststop":[]
}
```

The content of the **hook2.json** file is as follows:

```
# cat /etc/isulad-tools/hookspec.json
{
     "prestart": [
       {
            "path": "/docker-root/hooks/docker-hooks",
            "args": ["docker-hooks", "--state", "prestart"],
            "env": []
       }
      ],
      "poststart":[],
      "poststop":[
        {
            "path": "/docker-root/hooks/docker-hooks",
            "args": ["docker-hooks", "--state", "poststop"],
            "env": []
        }
       ]
}
```

The content in JSON format after manual combination is as follows:

```
{
      "prestart":[
        {
             "path": "/var/lib/docker/hooks/lxcfs-hook",
             "args": ["lxcfs-hook", "--log", "/var/log/lxcfs-hook.log"],
             "env": []
         },
         {
             "path": "/docker-root/hooks/docker-hooks",
             "args": ["docker-hooks", "--state", "prestart"],
             "env": []
         }
       ],
       "poststart":[],
       "poststop":[
           {
             "path": "/docker-root/hooks/docker-hooks",
             "args": ["docker-hooks", "--state", "poststop"],
             "env": []
           }
        ]
}
```

Docker daemon reads the binary values of hook in actions such as prestart in the hook configuration files in sequence based on the array sequence and executes the actions. Therefore, you need to identify the dependencies between multiple hooks. If required, you need to adjust the sequence of the hook configuration files according to the dependencies.

Customizing Default Hooks for All Containers

Docker daemon can receive the **--hook-spec** parameter. The semantics of **--hook-spec** is the same as that of **--hook-spec** in **docker create** or **docker run**. You can also add hook configurations to the **/etc/docker/daemon.json** file.

```
{
    "hook-spec": "/tmp/hookspec.json"
}
```

When a container is running, hooks specified in **--hook-spec** defined by daemon are executed first, and then hooks customized for each container are executed.

4.3.3 Configuring Health Check During Container Creation

Docker provides the user-defined health check function for containers. You can configure the **HEALTHCHECK CMD** option in the Dockerfile, or configure the **--health-cmd** option when a container is created so that commands are periodically executed in the container to monitor the health status of the container based on return values.

Configuration Methods

- Add the following configurations to the Dockerfile file:

  ```
  HEALTHCHECK --interval=5m --timeout=3s --health-exit-on-unhealthy=true \
    CMD curl -f http://localhost/ || exit 1
  ```

  The configurable options are as follows:

a. **--interval=DURATION**: interval between two consecutive command executions. The default value is **30s**. After a container is started, the first check is performed after the interval time.

b. **--timeout=DURATION**: maximum duration for executing a single check command. If the execution times out, the command execution fails. The default value is **30s**.

c. **--start-period=DURATION**: container initialization period. The default value is **0s**. During the initialization, the health check is also performed, while the health check failure is not counted into the maximum number of retries. However, if the health check is successful during initialization, the container is considered as started. All subsequent consecutive check failures are counted in the maximum number of retries.

d. **--retries=N**. maximum number of retries for the health check. The default value is **3**.

e. **--health-exit-on-unhealthy=BOOLEAN**: whether to kill a container when it is unhealthy. The default value is **false**.

f. **CMD**: This option is mandatory. If **0** is returned after a command is run in a container, the command execution succeeds. If a value other than **0** is returned, the command execution fails.

After **HEALTHCHECK** is configured, related configurations are written into the image configurations during image creation. You can run the **docker inspect** command to view the configurations. For example:

```
"Healthcheck": {
   "Test": [
      "CMD-SHELL",
      "/test.sh"
   ]
},
```

- Configurations during container creation:

```
docker run -itd --health-cmd "curl -f http://localhost/ || exit 1" --health-interval
5m --health-timeout 3s --health-exit-on-unhealthy centos bash
```

The configurable options are as follows:

a. **--health-cmd**: This option is mandatory. If **0** is returned after a command is run in a container, the command execution succeeds. If a value other than **0** is returned, the command execution fails.

b. **--health-interval**: interval between two consecutive command executions. The default value is **30s**. The upper limit of the value is the maximum value of Int64 (unit: nanosecond).

c. **--health-timeout**: maximum duration for executing a single check command. If the execution times out, the command execution fails. The default value is **30s**. The upper limit of the value is the maximum value of Int64 (unit: nanosecond).

d. **--health-start-period**: container initialization time. The default value is **0s**. The upper limit of the value is the maximum value of Int64 (unit: nanosecond).

e. **--health-retries**: maximum number of retries for the health check. The default value is **3**. The maximum value is the maximum value of Int32.

f. **--health-exit-on-unhealthy**: specifies whether to kill a container when it is unhealthy. The default value is **false**.

After the container is started, the **HEALTHCHECK** configurations are written into the container configurations. You can run the **docker inspect** command to view the configurations. For example:

```
"Healthcheck": {
   "Test": [
      "CMD-SHELL",
      "/test.sh"
   ]
},
```

Check Rules

1. After a container is started, the container status is **health:starting**.
2. After the period specified by **start-period**, the **cmd** command is periodically executed in the container at the interval specified by **interval**. That is, after the command is executed, the command will be executed again after the specified period.
3. If the **cmd** command is successfully executed within the time specified by **timeout** and the return value is **0**, the check is successful. Otherwise, the check fails. If the check is successful, the container status changes to **health:healthy**.
4. If the **cmd** command fails to be executed for the number of times specified by **retries**, the container status changes to **health:unhealthy**, and the container continues the health check.
5. When the container status is **health:unhealthy**, the container status changes to **health:healthy** if a check succeeds.
6. If **--health-exit-on-unhealthy** is set, and the container exits due to reasons other than being killed (the returned exit code is **137**), the health check takes effect only after the container is restarted.
7. When the **cmd** command execution is complete or times out, Docker daemon will record the start time, return value, and standard output of the check to the configuration file of the container. A maximum of five latest records can be recorded. In addition, the configuration file of the container stores health check parameters.

Run the **docker ps** command to view the container status.

```
[root@bac shm]# docker ps
CONTAINER ID       IMAGE             COMMAND           CREATED            STATUS
PORTS             NAMES
7de2228674a2       testimg           "bash"            About an hour ago   Up About an
hour (unhealthy)                    cocky_davinci
```

When the container is running, the health check status is written into the container configurations. You can run the **docker inspect** command to view the configurations.

```
"Health": {
   "Status": "healthy",
   "FailingStreak": 0,
   "Log": [
      {
         "Start": "2018-03-07T07:44:15.481414707-05:00",
         "End": "2018-03-07T07:44:15.556908311-05:00",
         "ExitCode": 0,
         "Output": ""
      },
      {
         "Start": "2018-03-07T07:44:18.557297462-05:00",
         "End": "2018-03-07T07:44:18.63035891-05:00",
         "ExitCode": 0,
         "Output": ""
```

```
        },
        ......
}
```

NOTE

- A maximum of five health check status records can be stored in a container. The last five records are saved.
- Only one health check configuration item can take effect in a container at a time. The later items configured in the Dockerfile will overwrite the earlier ones. Configurations during container creation will overwrite those in images.
- In the Dockerfile, you can set **HEALTHCHECK NONE** to cancel the health check configuration in a referenced image. When a container is running, you can set **--no-healthcheck** to cancel the health check configuration in an image. Do not configure the health check and **--no-healthcheck** parameters at the same time during the startup.
- After a container with configured health check parameters is started, if Docker daemon exits, the health check is not executed. After Docker daemon is restarted, the container health status changes to **starting**. Afterwards, the check rules are the same as above.
- If health check parameters are set to **0** during container image creation, the default values are used.
- If health check parameters are set to **0** during container startup, the default values are used.

4.3.4 Stopping and Deleting a Container

Run the **docker stop** command to stop the container named **container1**.

```
[root@localhost ~]# docker stop container1
```

Or run the **docker kill** command to kill and stop the container.

```
[root@localhost ~]# docker kill container1
```

After the container is stopped, run the **docker rm** command to delete the container.

```
[root@localhost ~]# docker rm container1
```

Or run the **docker rm -f** command to forcibly delete the container.

```
[root@localhost ~]# docker rm -f container1
```

Precautions

- Do not run the **docker rm –f** *XXX* command to delete a container. If you forcibly delete a container, the **docker rm** command ignores errors during the process, which may cause residual metadata of the container. If you delete an image in common mode and an error occurs during the deletion process, the deletion fails and no metadata remains.
- Do not run the **docker kill** command. The **docker kill** command sends related signals to service processes in a container. Depending on the signal processing policies of service processes in the container may cause the result that the signal execution cannot be performed as expected.
- A container in the restarting state may not stop immediately when you run the **docker stop** command. If a container uses the restart rules, when the container is in the restarting state, there is a low probability that the **docker stop** command on the container returns immediately. The container will still be restarted with the impact of the restart rule.
- Do not run the **docker restart** command to restart a container with the **--rm** parameter. When a container with the **--rm** parameter exits, the container is automatically deleted. If the container with the **--rm** parameter is restarted, exceptions may occur. For example, if both the **--rm** and **-ti** parameters are added when the container is started, the restart

operation cannot be performed on the container, otherwise, the container may stop responding and cannot exit.

When Using docker stop/restart to Specify -t and t<0, Ensure That Applications in the Container Can Process Stop Signal

Stop Principle: (The stop process is called by **Restart**.)

1. The SIGTERM (15) signal can be sent to a container for the first time.
2. Wait for a period of time (**t** entered by the user).
3. If the container process still exists, send the SIGKILL (9) signal to forcibly kill the process.

The meaning of the input parameter **t** (unit: s) is as follows:

- **t** < 0: Wait for graceful stop. This setting is preferred when users are assured that their applications have a proper stop signal processing mechanism.
- **t** = 0: Do not wait and send **kill -9** to the container immediately.
- **t** > 0: Wait for a specified period and send **kill -9** to the container if the container does not stop within the specified period.

Therefore, if **t** is set to a value less than 0 (for example, **t** = **-1**), ensure that the container application correctly processes the SIGTERM signal. If the container ignores this signal, the container will be suspended when the **docker stop** command is run.

Manually Deleting Containers in the Dead State As the Underlying File System May Be Busy

When Docker deletes a container, it stops related processes of the container, changes the container status to Dead, and then deletes the container rootfs. When the file system or devicemapper is busy, the last step of deleting rootfs fails. Run the **docker ps -a** command. The command output shows that the container is in the Dead state. Containers in the Dead state cannot be started again. Wait until the file system is not busy and run the **docker rm** command again to delete the containers.

In PID namespace Shared Containers, If Child Container Is in pause State, Parent Container Stops Responding and the docker run Command Cannot Be Executed

When the **--pid** parameter is used to create the parent and child containers that share PID namespace, if any process in the child container cannot exit (for example, it is in the D or pause state) when the **docker stop** command is executed, the **docker stop** command of the parent container is waiting. You need to manually recover the process so that the command can be executed normally.

In this case, run the **docker inspect** command on the container in the pause state to check whether the parent container corresponding to **PidMode** is the container that requires **docker stop**. For the required container, run the **docker unpause** command to cancel the pause state of the child container. Then, proceed to the next step.

Generally, the possible cause is that the PID namespace corresponding to the container cannot be destroyed due to residual processes. If the problem persists, use Linux tools to obtain the residual processes and locate the cause of the process exit failure in PID namespace. After the problem is solved, the container can exit.

- Obtain PID namespace ID in a container.

```
docker inspect --format={{.State.Pid}} CONTAINERID | awk '{print
"/proc/"$1"/ns/pid"}' |xargs readlink
```

- Obtain threads in the namespace.

```
 ls -l /proc/*/task/*/ns/pid |grep -F PIDNAMESPACE ID |awk '{print $9}' |awk -F  \/
'{print $5}'
```

4.3.5 Querying Container Information

In any case, the container status should not be determined based on whether the **docker** command is successfully returned. To view the container status, you are advised to use the following command:

```
docker inspect <NAME|ID>
```

4.3.6 Modification Operations

Precautions for Starting Multiple Processes in Container Using docker exec

When the first **docker exec** command executed in a container is the **bash** command, ensure that all processes started by **exec** are stopped before you run the **exit** command. Otherwise, the device may stop responding when you run the **exit** command. To ensure that the process started by **exec** is still running in the background when the **exit** command is run, add **nohup** when starting the process.

Usage Conflict Between docker rename and docker stats *container_name*

If you run the **docker stats** *container_name* command to monitor a container in real time, after the container is renamed by using **docker rename**, the name displayed after **docker stats** is executed is the original name instead of the renamed one.

Failed to Perform docker rename Operation on Container in restarting State

When the rename operation is performed on a container in the restarting state, Docker modifies the container network configuration accordingly. The container in the restarting state may not be started and the network does not exist. As a result, the rename operation reports an error indicating that the sandbox does not exist. You are advised to rename only containers that are not in the restarting state.

docker cp

1. When you run **docker cp** to copy files to a container, all operations on the container can be performed only after the **docker cp** command is executed.
2. When a container runs as a non-**root** user, and you run the **docker cp** command to copy a non-**root** file on the host to the container, the permission role of the file in the container changes to **root**. Different from the **cp** command, the **docker cp** command changes UIDs and GIDs of the files copied to the container to **root**.

docker login

After the **docker login** command is executed, **usrer/passwd** encrypted by AES (256-bit) is saved in **/root/.docker/config.json**. At the same time, *root***.docker/aeskey** (permission 0600) is generated to decrypt **usrer/passwd** in **/root/.docker/config.json**. Currently, AES key

cannot be updated periodically. You need to manually delete the AES key for updating. After AES key is updated, you need to log in to Docker daemon again to push the AES key no matter whether Docker daemon is restarted. For example:

```
root@hello:~/workspace/dockerfile# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have
a Docker ID, head over to https://hub.docker.com to create one.
Username: example Password:
Login Succeeded
root@hello:~/workspace/dockerfile# docker push example/empty
The push refers to a repository [docker.io/example/empty]
547b6288eb33: Layer already exists
latest: digest:
sha256:99d4fb4ce6c6f850f3b39f54f8eca0bbd9e92bd326761a61f106a10454b8900b size: 524
root@hello:~/workspace/dockerfile# rm /root/.docker/aeskey
root@hello:~/workspace/dockerfile# docker push example/empty
WARNING: Error loading config file:/root/.docker/config.json - illegal base64 data at
input byte 0
The push refers to a repository [docker.io/example/empty]
547b6288eb33: Layer already exists
errors:
denied: requested access to the resource is denied
unauthorized: authentication required
root@hello:~/workspace/dockerfile# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have
a Docker ID, head over to https://hub.docker.com to create one.
Username: example
Password:
Login Succeeded
root@hello:~/workspace/dockerfile# docker push example/empty
The push refers to a repository [docker.io/example/empty]
547b6288eb33: Layer already exists
latest: digest:
sha256:99d4fb4ce6c6f850f3b39f54f8eca0bbd9e92bd326761a61f106a10454b8900b size: 524
```

4.4 Image Management

4.4.1 Creating an Image

You can use the **docker pull**, **docker build**, **docker commit**, **docker import**, or **docker load** command to create an image. For details about how to use these commands, see 4.6.3 Image Management.

Precautions

1. Do not concurrently run the **docker load** and **docker rmi** commands. If both of the following conditions are met, concurrency problems may occur:
   - An image exists in the system.
   - The docker rmi and docker load operations are concurrently performed on an image.

Therefore, avoid this scenario. (All concurrent operations between the image creation operations such as running the **tag**, **build**, and **load**, and **rmi** commands, may cause similar errors. Therefore, do not concurrently perform these operations with **rmi**.)

2. If the system is powered off when docker operates an image, the image may be damaged. In this case, you need to manually restore the image.

   When the docker operates images (using the **pull**, **load**, **rmi**, **build**, **combine**, **commit**, or **import** commands), image data operations are asynchronous, and image metadata is synchronous. Therefore, if the system power is off when not all image data is updated to the disk, the image data may be inconsistent with the metadata. Users can view images (possibly none images), but cannot start containers, or the started containers are abnormal. In this case, run the **docker rmi** command to delete the image and perform the previous operations again. The system can be recovered.

3. Do not store a large number of images on nodes in the production environment. Delete unnecessary images in time.

   If the number of images is too large, the execution of commands such as **docker image** is slow. As a result, the execution of commands such as **docker build** or **docker commit** fails, and the memory may be stacked. In the production environment, delete unnecessary images and intermediate process images in time.

4. When the **--no-parent** parameter is used to build images, if multiple build operations are performed at the same time and the FROM images in the Dockerfile are the same, residual images may exist. There are two cases:

   - If FROM images are incomplete, the images generated when images of FROM are running may remain. Names of the residual images are similar to **base_v1.0.0-app_v2.0.0**, or they are none images.
   - If the first several instructions in the Dockerfile are the same, none images may remain.

None Image May Be Generated

1. A none image is the top-level image without a tag. For example, the image ID of **ubuntu** has only one tag **ubuntu**. If the tag is not used but the image ID is still available, the image ID becomes a none image.
2. An image is protected because the image data needs to be exported during image saving. However, if a deletion operation is performed, the image may be successfully untagged and the image ID may fail to be deleted (because the image is protected). As a result, the image becomes a none image.
3. If the system is powered off when you run the **docker pull** command or the system is in panic, a none image may be generated. To ensure image integrity, you can run the **docker rmi** command to delete the image and then restart it.
4. If you run the **docker save** command to save an image and specify the image ID as the image name, the loaded image does not have a tag and the image name is **none**.

A Low Probability That Image Fails to Be Built If the Image Is Deleted When Being Built

Currently, the image build process is protected by reference counting. After an image is built, reference counting of the image is increased by 1 (holdon operation). Once the holdon operation is successful, the image will not be deleted. However, there is a low probability that before the holdon operation is performed, the image can still be deleted, causing the image build failure.

4.4.2 Viewing Images

Run the following command to view the local image list:

```
docker images
```

4.4.3 Deleting Images

Precautions

Do not run the **docker rmi –f** *XXX* command to delete images. If you forcibly delete an image, the **docker rmi** command ignores errors during the process, which may cause residual metadata of containers or images. If you delete an image in common mode and an error occurs during the deletion process, the deletion fails and no metadata remains.

4.5 Command Reference

4.5.1 Container Engine

Docker daemon is a system process that resides in the background. Before you run a docker subcommand, start Docker daemon.

If the Docker daemon is installed using the RPM package or system package management tool, you can run the **systemctl start docker** command to start the Docker daemon.

The **docker** command supports the following parameters:

1. To combine parameters of a single character, run the following command:

   ```
   docker run -t -i busybox /bin/sh
   ```

   The command can be written as follows:

   ```
   docker run -ti busybox /bin/sh
   ```

2. **bool** command parameters such as **--icc=true**, are displayed in the command help. If this parameter is not used, the default value displayed in the command help is used. If this parameter is used, the opposite value of the value displayed in the command help is used. In addition, if **--icc** is not added when Docker daemon is started, **--icc=true** is used by default. Otherwise, **--icc=false** is used.
3. Parameters such as **--attach=[]** in the command help indicate that these parameters can be set for multiple times. For example:

   ```
   docker run --attach=stdin --attach=stdout -i -t busybox /bin/sh
   ```

4. Parameters such as **-a** and **--attach=[]** in the command help indicate that the parameter can be specified using either **-a** *value* or **--attach=***value*. For example:

   ```
   docker run -a stdin --attach=stdout -i -t busybox /bin/sh
   ```

5. Parameters such as **--name=""** can be configured with a character string and can be configured only once. Parameters such as **-c=** can be configured with an integer and can be configured only once.

Table 4-1 Parameters specified during the Docker daemon startup

Parameter	Description
--api-cors-header	CORS header information for enabling remote API calling. This interface supports the secondary development of upper-layer applications, which sets the CORS header for a remote API.
--authorization-plugin=[]	Authentication plug-in.
-b, --bridge=""	Existing bridge device mounting to the docker container. Note: **none** can be used to disable the network in the container.
--bip=""	Bridge IP address, which is automatically created using the CIDR address. Note: this parameter cannot be used with **-b** .
--cgroup-parent	cgroup parent directory configured for all containers.
--config-file=/etc/docker/daemon.json	Configuration file for starting Docker daemon.
--containerd	Socket path of containerd.
-D, --debug=false	Specifies whether to enable the debugging mode.
--default-gateway	Default gateway of the container IPv4 address.
--default-gateway-v6	Default gateway of the container IPv6 address.
--default-ulimit=[]	Default ulimit value of the container.
--disable-legacy-registry	Disables the original registry.
--dns=[]	DNS server of the forcibly used container. Example: **--dns 8.8.***x.x*
--dns-opt=[]	DNS option.
--dns-search=[]	Forcibly searches DNS search domain name used by a container. Example: **--dns-search example.com**

--exec-opt=[]	Parameter to be executed when a container is started.
	For example, set the **native.umask** parameter.
	`#The umask value of the started container is 0022.`
	`--exec-opt native.umask=normal`
	
	`#The umask value of the started container is 0027 (default value).`
	`--exec-opt native.umask=secure`
	Note: If **native.umask** is also configured in **docker create** or **docker run** command, the configuration in command is used.

--exec-root=/var/run/docker	Root directory for storing the execution status file.

Parameter	Description
--fixed-cidr=""	Fixed IP address (for example, **10.20.0.0/16**) of the subnet. The IP address of the subnet must belong to the network bridge.
--fixed-cidr-v6	Fixed IPv6 address.
-G, --group="docker"	Group assigned to the corresponding Unix socket in the background running mode. Note: When an empty string is configured for this parameter, the group information is removed.
-g, --graph="/var/lib/docker"	The root directory for running docker.
-H, --host=[]	Socket bound in background mode. One or more sockets can be configured using **tcp://***host***:***port***, unix:///***path to socket***, fd://*** or **fd://socketfd**. Example: **$ dockerd -H tcp://0.0.0.0:2375** or **$ export DOCKER_HOST="tcp://0.0.0.0:2375"**
--insecure-registry=[]	Registry for insecure connections. By default, the Docker uses TLS certificates to ensure security for all connections. If the registry does not support HTTPS connections or the certificate is issued by an unknown certificate authority of the Docker daemon, you need to configure **--insecure-registry=192.168.1.110:5000** when starting the daemon. This parameter needs to be configured if a private registry is used.
--image-layer-check=true	Image layer integrity check. To enable the function, set this parameter to **true**. Otherwise, set this parameter to **false**. If this parameter is not configured, the function is disabled by default. When Docker is started, the image layer integrity is checked. If the image layer is damaged, the related images are unavailable. Docker cannot verify empty files, directories, or link files. Therefore, if the preceding files are lost due to a power failure, the integrity check of Docker image data may fail. When the Docker version changes, check whether the parameter is supported. If not supported, delete it from the configuration file.
--icc=true	Enables communication between containers.
--ip="0.0.0.0"	Default IP address used when a container is bound to a port.
--ip-forward=true	Starts the net.ipv4.ip_forward process of the container.
--ip-masq=true	Enables IP spoofing.

Parameter	Description
--iptables=true	Starts the iptables rules defined by the Docker container.
-l, --log-level=info	Log level.
--label=[]	Daemon label, in *key=value* format.
--log-driver=json-file	Default log driver of container logs.
--log-opt=map[]	Log drive parameters.
--mtu=0	MTU value of the container network. If this parameter is not configured, value of **route MTU** is used by default. If the default route is not configured, set this parameter to the constant value **1500**.
-p, --pidfile="/var/run/docker.pid"	PID file path of the background process.
--raw-logs	Logs with all timestamps and without the ANSI color scheme.
--registry-mirror=[]	Image registry preferentially used by the dockerd.
-s, --storage-driver=""	Storage driver used when a container is forcibly run.
--selinux-enabled=false	Enables SELinux. If the kernel version is 3.10.0-862.14 or later, this parameter cannot be set to **true**.
--storage-opt=[]	Storage driver parameter. This parameter is valid only when the storage driver is devicemapper. Example: **dockerd --storage-opt dm.blocksize=512K**
--tls=false	Enables the TLS authentication.
--tlscacert="/root/.docker/ca.pem"	Certificate file path that has been authenticated by the CA.
--tlscert="/root/.docker/cert.pem"	File path of the TLS certificates.
--tlskey="/root/.docker/key.pem"	File path of TLS keys.
--tlsverify=false	Verifies the communication between the background processes and the client using TLS.
--insecure-skip-verify-enforce	Whether to forcibly skip the verification of the certificate host or domain name. The default value is **false**.
--use-decrypted-key=true	Whether to use the decryption private key.
--userland-proxy=true	Whether to use the userland proxy for the container LO device.
--userns-remap	User namespace-based user mapping table in the container.

Parameter	Description
	NOTE This parameter is not supported in the current version.

4.5.2 Container Management

Subcommands supported by the current Docker are classified into the following groups by function:

Function	Command		Description
Host environment	version		Views the Docker version.
	info		Views the Docker system and host environment information.
Container-related information	Container lifecycle management	create	Creates a container using an image.
		run	Creates and runs a container using an image.
		start	Starts a stopped container.
		stop	Stops a running container.
		restart	Restarts a container.
		wait	Waits for a container to stop and prints the exit code.
		rm	Deletes a container.
	Container process management	pause	Suspends all processes in a container.
		unpause	Resumes a suspended process in a container.
		top	Views processes in a container.
		exec	Executes a process in containers.
	Container inspection	ps	Views running containers (without

Function	Command		Description
	tool		attaching any option).
		logs	Displays the log information of a container.
		attach	Connects standard input and output to a container.
		inspect	Returns the bottom-layer information of a container.
		port	Lists the port mappings between containers and hosts.
		diff	Returns the changes made by the container compared with rootfs in the image.
		cp	Copies files between containers and hosts.
		export	Exports the file system in a container in a .tar package.
		stats	Views the resource usage of a container in real time.
Images	Generates an image.	build	Creates an image using a Dockerfile.
		commit	Creates an image based on the container rootfs.
		import	Creates an image using the content in the .tar package as the file system.
		load	Loads an image from the .tar package.
	Image registry	login	Logs in to a registry.
		logout	Logs out of a

Function	Command		Description
			registry.
		pull	Pulls an image from the registry.
		push	Pushes an image to the registry.
		search	Searches for an image in the registry.
	Image management	images	Displays images in the system.
		history	Displays the change history of an image.
		rmi	Deletes an image.
		tag	Adds a tag to an image.
		save	Saves an image to a .tar package.
Others	events		Obtains real-time events from the Docker daemon.
	rename		Renames a container.

Some subcommands have some parameters, such as **docker run**. You can run the **docker** *command* **--help** command to view the help information of the command. For details about the command parameters, see the preceding command parameter description. The following sections describe how to use each command.

4.5.2.1 attach

Syntax: **docker attach [***options***]** *container*

Function: Attaches an option to a running container.

Parameter description:

--no-stdin=false: Does not attach any STDIN.

--sig-proxy=true: Proxies all signals of the container, except SIGCHLD, SIGKILL, and SIGSTOP.

Example:

```
$ sudo docker attach attach test
root@2988b8658669:/# ls bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc
root  run  sbin  srv  sys  tmp  usr  var
```

4.5.2.2 commit

Syntax: **docker commit [***options***]** *container* **[***repository[:tag]***]**

Function: creates an image from a container.

Parameter description:

-a, **--author=""**: specifies an author.

-m, **--message=""**: specifies the submitted information.

-p, **--pause=true**: pauses the container during submission.

Example:

Run the following command to start a container and submit the container as a new image:

```
$ sudo docker commit test busybox:test
sha256:be4672959e8bd8a4291fbdd9e99be932912fe80b062fba3c9b16ee83720c33e1

$ sudo docker images
REPOSITORY         TAG              IMAGE ID          CREATED           SIZE
busybox           latest            e02e811dd08f       2 years ago        1.09MB
```

4.5.2.3 cp

Syntax: **docker cp [***options***]** *container***:***src_path dest_path***|-**

docker cp [*options***]** *src_path***|-** *container***:***dest_path*

Function: Copies a file or folder from a path in a container to a path on the host or copies a file or folder from the host to the container:

Precautions: The **docker cp** command does not support the copy of files in virtual file systems such as **/proc**, **/sys**, **/dev**, and **/tmp** in the container and files in the file systems mounted by users in the container.

Parameter description:

-a, **--archive**: Sets the owner of the file copied to the container to the **container** user (**--user**).

-L, **--follow-link**: Parses and traces the symbolic link of a file.

Example:

Run the following command to copy the **/test** directory in the registry container to the **/home/***aaa* directory on the host:

```
$ sudo docker cp registry:/test /home/aaa
```

4.5.2.4 create

Syntax: **docker create [***options***]** *image* **[***command***] [***arg***...]**

Function: Creates a container using an image file and return the ID of the container. After the container is created, run the **docker start** command to start the container. *options* are used to configure the container during container creation. Some parameters will overwrite the container configuration in the image file. *command* indicates the command to be executed during container startup.

Parameter description:

Table 4-2 Parameter description

Parameter	Description
-a --attach=[]	Attaches the console to the STDIN, STDOUT, and STDERR of the process in the container.
--name=""	Name of a container.
--add-host=[host:ip]	Adds a mapping between the host name and IP address to the **/etc/hosts** in the container. For example, **--add-host=test:10.10.10.10**.

--annotation	Sets annotations for the container. For example, set the **native.umask** parameter.
	`--annotation native.umask=normal #The umask value of the started container is 0022.`
	`--annotation native.umask=secure #The umask value of the started container is 0027.`
	If this parameter is not set, the **umask** configuration in dockerd is used.

--blkio-weight	Relative weight of blockio, which ranges from 10 to 1000.
--blkio-weight-device=[]	Blockio weight, which configures the relative weight.
-c, --cpu-shares=0	Relative weight of the host CPU obtained by the container. This parameter can be used to obtain a higher priority. By default, all containers obtain the same CPU priority.
--cap-add=[]	Adds Linux functions.
--cap-drop=[]	Clears Linux functions.
--cgroup-parent	cgroup parent directory for the container.
--cidfile=""	Writes the container ID to a specified file. For example: **--cidfile=/home/cidfile-test** writes the container ID to the **/home/cidfile-test** file.
--cpu-period	CPU CFS period. The default value is **100** ms. Generally, **--cpu-period** and **--cpu-quota** are used together. For example, **--cpu-period=50000 --cpu-quota=25000** indicates that if there is one CPU, the container can obtain 50% of the CPU every 50 ms. **--cpus=0.5** has the same effect.
--cpu-quota	CPU CFS quota. The default value is **0**, indicating that there is no restriction on the quota.
--cpuset-cpus	Number of CPUs (0-3, 0, 1) that can be used by processes in the container. By default, there is no restriction on this parameter.

Parameter	Description
--cpuset-mems	Memory nodes (0-3, 0, 1) for running processes in the container. This parameter is valid only for the NUMA system.
--device=[]	Adds the host device to a container, for example, **--device=/dev/sdc:/dev/xvdc:rwm**.
--dns=[]	Forcibly enables the container to use the specified DNS server. For example, **--dns=114.114.***xxx.xxx* indicates that **nameserver 114.114.***xxx.xxx* is written to **/etc/resolv.conf** of the created container and the original content is overwritten.
--dns-opt=[]	DNS options.
--dns-search=[]	Forcibly searches DNS search domain name used by a container.
-e, --env=[]	Sets environment variable for the container. --env=[KERNEL_MODULES=]: Inserts a specified module into a container. Currently, only the modules on the host can be inserted. After the container is deleted, the modules still reside on the host, and the **--hook-spec** option must be configured for the container. The following are valid parameter formats: KERNEL_MODULERS= KERNEL_MODULERS=a KERNEL_MODULERS=a,b KERNEL_MODULERS=a,b,
--entrypoint=""	Overwrites the original **entrypoint** in the image. The **entrypoint** is used to set the command executed when the container is started.
--env-file=[]	Reads environment variables from a file. Multiple environment variables are separated by lines in the file. For example: **--env-file=/home/test/env** indicates multiple environment variables are stored in the **env** file.
--expose=[]	Enables an internal port of a container. The **-P** option described in the following section maps the enabled port to a port on the host.
--group-add=[]	Adds a specified container to an additional group.
-h, --hostname=""	Host name.
--health-cmd	Container health check command.
--health-interval	Interval between two consecutive command executions. The default value is **30s**.
--health-timeout	Maximum duration for executing a single check command. If the execution times out, the command fails to be executed. The default value is **30s**.

Parameter	Description
--health-start-period	Interval between the time when the container is started and the time when the first health check is performed. The default value is **0s**.
--health-retries	Maximum number of retries after a health check fails. The default value is **3**.
--health-exit-on-unhealthy	Specifies whether to stop a container when the container is unhealthy. The default value is **false**.
--host-channel=[]	Sets a channel for communication between processes in the container and the host, in *host path*:*container path*:*rw/ro*:*size limit* format.
-i, --interactive=false	Enables STDIN even if it is not attached.
--ip	IPv4 address of a container.
--ip6	IPv6 address of a container.
--ipc	IPC namespace of a container.
--isolation	Container isolation policy.
-l, --label=[]	Label of a container.
--label-file=[]	Obtains the label from the file.
--link=[]	Links to another container. This parameter adds environment variables of the IP address and port number of the linked container to the container and adds a mapping to the **/etc/hosts** file, for example, **--link=***name:alias*.
--log-driver	Log driver of a container.
--log-opt=[]	Log driver option.
-m, --memory=""	Memory limit of a container. The format is *numberoptional unit*, and available units are **b**, **k**, **m**, and **g**. The minimum value of this parameter is **4m**.
--mac-address	MAC address of a container, for example, **92:d0:c6:0a:***xx***:***xx*.
--memory-reservation	Container memory limit. The default value is the same as that of **--memory**. **--memory** is a hard limit, and **--memory-reservation** is a soft limit. When the memory usage exceeds the preset value, the memory usage is dynamically adjusted (the system attempts to reduce the memory usage to a value less than the preset value when reclaiming the memory). However, the memory usage may exceed the preset value. Generally, this parameter can be used together with **--memory**. The value must be less than the preset value of **--memory**.
--memory-swap	Total usage of the common memory and swap partition. **-1** indicates no restriction is set on the usage. If this parameter is not set, the swap partition size is twice the value of

Parameter	Description
	--memory. That is, the swap partition can use the same amount of memory as **--memory**.
--memory-swappiness=-1	Time when the container uses the swap memory. The value ranges from 0 to 100, in percentage.
--net="bridge"	Network mode of the container. Docker 1.3.0 has the following network modes: **bridge**, **host**, **none**, and **container:***name*|*id*. The default value is **bridge**. • **bridge**: Creates a network stack on the bridge when the Docker daemon is started. • **host**: Uses the network stack of the host in the container. • **none**: Does not use networks. • **container:***name*|*id*: Reuses the network stack of another container.
--no-healthcheck	Does not perform health check for a container.
--oom-kill-disable	Disables the OOM killer. You are advised not to set this parameter if the **-m** parameter is not set.
--oom-score-adj	Adjusts the OOM rule of a container. The value ranges from -1000 to 1000.
-P, --publish-all=false	Maps all enabled ports of a container to host ports. Containers can be accessed through the host ports. You can run the **docker port** command to view the mapping between container ports and host ports.
-p, --publish=[]	Maps a port in a container to a port on the host, in *IP address*:*host port*:*container port* | *IP address*::*container port* | *host port*:*container port* | *container port* format. If no IP address is configured, accesses of all NICs on the host is listened. If no host port is configured, the host port is automatically allocated.
--pid	PID namespace of a container.
--privileged=false	Grants extra permission to a container. If the **--privileged** option is used, the container can access all devices on the host.
--restart=""	Configures restart rule when the container exits. Currently, version 1.3.1 supports the following rules: • **no**: indicates that the container is not restarted when it is stopped. • **on-failure**: indicates that the container is restarted when the container exit code is not 0. This rule can be used to add the maximum number of restart times, for example, **on-failure:5**, indicating that the container can be restarted for a maximum of five times. • **always**: indicates the container is exited regardless of the exit code.

Parameter	Description
--read-only	Mounts the root file system of the container in read-only mode.
--security-opt=[]	Container security rule.
--shm-size	Size of the **/dev/shm** device. The default value is **64M**.
--stop-signal=SIGTERM	Container stop signal. The default value is **SIGTERM**.
-t, --tty=false	Allocates a pseudo terminal.
--tmpfs=[]	Mounts the tmpfs directory.
-u, --user=""	User name or user ID.
--ulimit=[]	ulimit option.
--userns	User namespace of a container.
-v, --volume=[]	Mounts a directory of the host to the container, or create a volume in the container. For example, **-v /home/test:/home** mounts the **/home/test** directory of the host to the **/home** directory of the container, and **-v /tmp** creates the **tmp** folder in the **root** directory of the container, the folder can be shared by other containers using the **--volumes-from** option. The host directory cannot be mounted to the **/proc** subdirectory of the container. Otherwise, an error is reported when the container is started.
--volume-driver	Data volume driver of the container. This parameter is optional.
--volumes-from=[]	Mounts the volume of another container to the current container to share the volume. For example, **-volumes-from** *container_name* mounts the volume of *container_name* to the current container. **-v** and **--volumes-from=[]** are two very important options for data backup and live migration.
-w, --workdir=""	Specifies the working directory of the container.

Example:

Run the following command to create a container named **busybox** and run the **docker start** command to start the container.

```
$ sudo docker create -ti --name=busybox busybox /bin/bash
```

4.5.2.5 diff

Syntax: **docker diff** *container*

Function: Checks the differences between containers and determines the changes have been made compared with the container creation.

Parameter description: none.

Example:

```
$ sudo docker diff registry
C /root
A /root/.bash history
A /test
```

4.5.2.6 exec

Syntax: **docker exec [***options***]** *container command* **[***arg...***]**

Function: Runs a command in the container.

Parameter description:

-d and **--detach=false**: Run in the background.

-i and **--interactive=false**: Keep the STDIN of the container enabled.

-t and **--tty=false**: Allocate a virtual terminal.

--privileged: Executes commands in privilege mode.

-u and **--user**: Specifies the user name or UID.

Example:

```
$ sudo docker exec -ti exec test ls
bin  etc   lib      media  opt   root  sbin  sys   tmp  var
dev  home  lib64  mnt   proc  run   srv   test  usr
```

4.5.2.7 export

Syntax: **docker export** *container*

Function: Exports the file system content of a container to STDOUT in .tar format.

Parameter description: none.

Example:

Run the following commands to export the contents of the container named **busybox** to the **busybox.tar** package:

```
$ sudo docker export busybox > busybox.tar
$ ls
busybox.tar
```

4.5.2.8 inspect

Syntax: **docker inspect [***options***]** *container*|*image* **[***container*|*image...***]**

Function: Returns the underlying information about a container or image.

Parameter description:

-f and **--format=""**: Output information in a specified format.

-s and **--size**: Display the total file size of the container when the query type is container.

--type: Returns the JSON format of the specified type.

-t and **--time=120**: Timeout interval, in seconds. If the **docker inspect** command fails to be executed within the timeout interval, the system stops waiting and immediately reports an error. The default value is **120**.

Example:

1. Run the following command to return information about a container:

```
$ sudo docker inspect busybox test
[
    {
        "Id": "9fbb8649d5a8b6ae106bb0ac7686c40b3cbd67ec2fd1ab03e0c419a70d755577",
        "Created": "2019-08-28T07:43:51.27745746Z",
        "Path": "bash",
        "Args": [],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 64177,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2019-08-28T07:43:53.021226383Z",
            "FinishedAt": "0001-01-01T00:00:00Z"
        },
......
```

2. Run the following command to return the specified information of a container in a specified format. The following uses the IP address of the busybox_test container as an example.

```
$ sudo docker inspect -f {{.NetworkSettings.IPAddress}} busybox test
172.17.0.91
```

4.5.2.9 logs

Syntax: **docker logs [***options***]** *container*

Function: Captures logs in a container that is in the **running** or **stopped** state.

Parameter description:

-f and **--follow=false**: Print logs in real time.

-t and **--timestamps=false**: Display the log timestamp.

--since: Displays logs generated after the specified time.

--tail="all": Sets the number of lines to be displayed. By default, all lines are displayed.

Example:

1. Run the following command to check the logs of the jaegertracing container where a jaegertracing service runs:

```
$ sudo docker logs jaegertracing
{"level":"info","ts":1566979103.3696961,"caller":"healthcheck/handler.go:99","m
sg":"Health Check server started","http-port":14269,"status":"unavailable"}
{"level":"info","ts":1566979103.3820567,"caller":"memory/factory.go:55","msg":"
Memory storage configuration","configuration":{"MaxTraces":0}}
{"level":"info","ts":1566979103.390773,"caller":"tchannel/builder.go:94","msg":
"Enabling service discovery","service":"jaeger-collector"}
{"level":"info","ts":1566979103.3908608,"caller":"peerlistmgr/peer list mgr.go:
111","msg":"Registering active peer","peer":"127.0.0.1:14267"}
{"level":"info","ts":1566979103.3922884,"caller":"all-in-one/main.go:186","msg"
:"Starting agent"}
{"level":"info","ts":1566979103.4047635,"caller":"all-in-one/main.go:226","msg"
:"Starting jaeger-collector TChannel server","port":14267}
{"level":"info","ts":1566979103.404901,"caller":"all-in-one/main.go:236","msg":
"Starting jaeger-collector HTTP server","http-port":14268}
{"level":"info","ts":1566979103.4577134,"caller":"all-in-one/main.go:256","msg"
:"Listening for Zipkin HTTP traffic","zipkin.http-port":9411}
```

2. Add **-f** to the command to output the logs of the jaegertracing container in real time.

```
$ sudo docker logs -f jaegertracing
{"level":"info","ts":1566979103.3696961,"caller":"healthcheck/handler.go:99","m
sg":"Health Check server started","http-port":14269,"status":"unavailable"}
{"level":"info","ts":1566979103.3820567,"caller":"memory/factory.go:55","msg":"
Memory storage configuration","configuration":{"MaxTraces":0}}
{"level":"info","ts":1566979103.390773,"caller":"tchannel/builder.go:94","msg":
"Enabling service discovery","service":"jaeger-collector"}
{"level":"info","ts":1566979103.3908608,"caller":"peerlistmgr/peer list mgr.go:
111","msg":"Registering active peer","peer":"127.0.0.1:14267"}
{"level":"info","ts":1566979103.3922884,"caller":"all-in-one/main.go:186","msg"
:"Starting agent"}
```

4.5.2.10 pause/unpause

Syntax: **docker pause** *container*

docker unpause *container*

Function: The two commands are used in pairs. The **docker pause** command suspends all processes in a container, and the **docker unpause** command resumes the suspended processes.

Parameter description: none.

Example:

The following uses a container where the docker registry service runs as an example. After the **docker pause** command is executed to pause the process of the container, access of the registry service by running the **curl** command is blocked. You can run the **docker unpause** command to resume the suspended registry service. The registry service can be accessed by running the **curl** command.

1. Run the following command to start a registry container:

```
$ sudo docker run -d --name pause_test -p 5000:5000 registry
```

Run the **curl** command to access the service. Check whether the status code **200 OK** is returned.

```
$ sudo curl -v 127.0.0.1:5000
```

2. Run the following command to stop the processes in the container:

```
$ sudo docker pause pause_test
```

Run the **curl** command to access the service to check whether it is blocked and wait until the service starts.

3. Run the following command to resume the processes in the container:

```
$ sudo docker unpause pause_test
```

The cURL access in step 2 is restored and the request status code **200 OK** is returned.

4.5.2.11 port

Syntax: **docker port** *container* **[***private_port[/proto]***]**

Function: Lists the port mapping of a container or queries the host port where a specified port resides.

Parameter description: none.

Example:

1. Run the following command to list all port mappings of a container:

```
$ sudo docker port registry
5000/tcp -> 0.0.0.0.:5000
```

2. Run the following command to query the mapping of a specified container port:

```
$ sudo docker port registry 5000
0.0.0.0.:5000
```

4.5.2.12 ps

Syntax: **docker ps [***options***]**

Function: Lists containers in different states based on different parameters. If no parameter is added, all running containers are listed.

Parameter description:

-a and **--all=false**: Display the container.

-f and **--filter=[]**: Filter values. The available options are: **exited=***int* (exit code of the container) **status=***restarting*|*running*|*paused*|*exited* (status code of the container), for example, **-f status=running**: lists the running containers.

-l and **--latest=false**: List the latest created container.

-n=-1: Lists the latest created *n* containers.

--no-trunc=false: Displays all 64-bit container IDs. By default, 12-bit container IDs are displayed.

-q and **--quiet=false**: Display the container ID.

-s and **--size=false**: Display the container size.

Example:

1. Run the following command to lists running containers:

```
$ sudo docker ps
```

2. Run the following command to display all containers:

```
$ sudo docker ps -a
```

4.5.2.13 rename

Syntax: **docker rename OLD_NAME NEW_NAME**

Function: Renames a container.

Example:

Run the **docker run** command to create and start a container, run the **docker rename** command to rename the container, and check whether the container name is changed.

```
$ sudo docker ps
CONTAINER ID       IMAGE             COMMAND           CREATED           STATUS
PORTS             NAMES
b15976967abb       busybox:latest       "bash"            3 seconds ago      Up 2
seconds                        festive morse
$ sudo docker rename pedantic euler new name
$ sudo docker ps
CONTAINER ID       IMAGE             COMMAND           CREATED           STATUS
PORTS             NAMES
b15976967abb       busybox:latest       "bash"            34 seconds ago     Up 33
seconds                       new_name
```

4.5.2.14 restart

Syntax: **docker restart [***options***]** *container* **[***container...***]**

Function: Restarts a running container.

Parameter description:

-t and **--time=10**: Number of seconds to wait for the container to stop before the container is killed. If the container has stopped, restart the container. The default value is **10**.

Example:

```
$ sudo docker restart busybox
```

NOTE

During the container restart, if a process in the **D** or **Z** state exists in the container, the container may fail to be restarted. In this case, you need to analyze the cause of the **D** or **Z** state of the process in the container. Restart the container after the **D** or **Z** state of the process in the container is released.

4.5.2.15 rm

Syntax: **docker rm [***options***]** *container* **[***container...***]**

Function: Deletes one or more containers.

Parameter description:

-f and **--force=false**: Forcibly delete a running container.

-l and **--link=false**: Remove the specified link and do not remove the underlying container.

-v and **--volumes=false**: Remove the volumes associated with the container.

Example:

1. Run the following command to delete a stopped container:

```
$ sudo docker rm test
```

2. Run the following command to delete a running container:

```
$ sudo docker rm -f rm_test
```

4.5.2.16 run

Syntax: **docker run [***options***] *image* [***command***] [***arg***...]**

Function: Creates a container from a specified image (if the specified image does not exist, an image is downloaded from the official image registry), starts the container, and runs the specified command in the container. This command integrates the **docker create**, **docker start**, and **docker exec** commands.

Parameter description: (The parameters of this command are the same as those of the **docker create** command. For details, see the parameter description of the **docker create** command. Only the following two parameters are different.)

--rm=false: Specifies the container to be automatically deleted when it exits.

-v: Mounts a local directory or an anonymous volume to the container. Note: When a local directory is mounted to a container with a SELinux security label, do not add or delete the local directory at the same time. Otherwise, the security label may not take effect.

--sig-proxy=true: Receives proxy of the process signal. SIGCHLD, SIGSTOP, and SIGKILL do not use the proxy.

Example:

Run the busybox image to start a container and run the **/bin/sh** command after the container is started:

```
$ sudo docker run -ti busybox /bin/sh
```

4.5.2.17 start

Syntax: **docker start [***options***] *container* [***container***...]**

Function: Starts one or more containers that are not running.

Parameter description:

-a and **--attach=false**: Attach the standard output and error output of a container to STDOUT and STDERR of the host.

-i and **--interactive=false**: Attach the standard input of the container to the STDIN of the host.

Example:

Run the following command to start a container named **busybox** and add the **-i -a** to the command to add standard input and output. After the container is started, directly enter the container. You can exist the container by entering **exit**.

If **-i -a** is not added to the command when the container is started, the container is started in the background.

```
$ sudo docker start -i -a busybox
```

4.5.2.18 stats

Syntax: **docker stats [***options***] [***container***...]**

Function: Continuously monitors and displays the resource usage of a specified container. (If no container is specified, the resource usage of all containers is displayed by default.)

Parameter description:

-a, and **--all**: Display information about all containers. By default, only running containers are displayed.

--no-stream: Displays only the first result and does not continuously monitor the result.

Example:

Run the **docker run** command to start and create a container, and run the **docker stats** command to display the resource usage of the container:

```
$ sudo docker stats
CONTAINER ID       NAME                 CPU %             MEM USAGE / LIMIT    MEM %
NET I/O           BLOCK I/O          PIDS
2e242bcdd682       jaeger               0.00%             77.08MiB / 125.8GiB   0.06%
42B / 1.23kB       97.9MB / 0B        38
02a06be42b2c       relaxed chandrasekhar   0.01%             8.609MiB / 125.8GiB
0.01%             0B / 0B           0B / 0B           10
deb9e49fdef1       hardcore montalcini     0.01%             12.79MiB / 125.8GiB
0.01%             0B / 0B           0B / 0B           9
```

4.5.2.19 stop

Syntax: **docker stop [***options***]** *container* **[***container***...]**

Function: Sends a SIGTERM signal to a container and then sends a SIGKILL signal to stop the container after a certain period.

Parameter description:

-t and **--time=10**: Number of seconds that the system waits for the container to exit before the container is killed. The default value is **10**.

Example:

```
$ sudo docker stop -t=15 busybox
```

4.5.2.20 top

Syntax: **docker top** *container* **[***ps options***]**

Function: Displays the processes running in a container.

Parameter description: none.

Example:

Run the top_test container and run the **top** command in the container.

```
$ sudo docker top top test
UID                 PID                 PPID                C                   STIME
TTY                 TIME                CMD
root                70045               70028               0                   15:52
pts/0               00:00:00            bash
```

The value of **PID** is the PID of the process in the container on the host.

4.5.2.21 update

Syntax: **docker update [***options***]** *container* **[***container***...]**

Function: Hot changes one or more container configurations.

Parameter description:

Table 4-3 Parameter description

Parameter	Description
--accel=[]	Configures one or more container accelerators.
--blkio-weight	Relative weight of the container blockio. The value ranges from 10 to 1000.
--cpu-shares	Relative weight of the host CPU obtained by the container. This parameter can be used to obtain a higher priority. By default, all containers obtain the same CPU priority.
--cpu-period	CPU CFS period. The default value is **100** ms. Generally, **--cpu-period** and **--cpu-quota** are used together. For example, **--cpu-period=50000 --cpu-quota=25000** indicates that if there is one CPU, the container can obtain 50% of the CPU every 50 ms.
--cpu-quota	CPU CFS quota. The default value is **0**, indicating that there is no restriction on the quota.
--cpuset-cpus	Number of CPUs (0-3, 0, 1) that can be used by processes in the container. By default, there is no restriction on this parameter.
--cpuset-mems	Memory nodes (0-3, 0, 1) for running processes in the container. This parameter is valid only for the NUMA system.
--kernel-memory=""	Kernel memory limit of a container. The format is *numberoptional unit*, and available units are **b**, **k**, **m**, and **g**.
-m, --memory=""	Memory limit of a container. The format is *numberoptional unit*, and available units are **b**, **k**, **m**, and **g**. The minimum value of this parameter is **4m**.

Parameter	Description
--memory-reservation	Container memory limit. The default value is the same as that of **--memory**. **--memory** is a hard limit, and **--memory-reservation** is a soft limit. When the memory usage exceeds the preset value, the memory usage is dynamically adjusted (the system attempts to reduce the memory usage to a value less than the preset value when reclaiming the memory). However, the memory usage may exceed the preset value. Generally, this parameter can be used together with **--memory**. The value must be less than the preset value of **--memory**.
--memory-swap	Total usage of the common memory and swap partition. **-1** indicates no restriction is set on the usage. If this parameter is not set, the swap partition size is twice the value of **--memory**. That is, the swap partition can use the same amount of memory as **--memory**.
--restart=""	Configures restart rule when the container exits. Currently, version 1.3.1 supports the following rules: • **no**: indicates that the container is not restarted when it is stopped. • **on-failure**: indicates that the container is restarted when the container exit code is not 0. This rule can be used to add the maximum number of restart times, for example, **on-failure:5**, indicating that the container can be restarted for a maximum of five times. • **always**: indicates the container is exited regardless of the exit code.
--help	Help information.

Example:

Run the following command to change the CPU and memory configurations of the container named **busybox**, including changing the relative weight of the host CPU obtained by the container to **512**, the CPU cores that can be run by processes in the container to **0,1,2,3**, and the memory limit for running the container to **512 m**.

```
$ sudo docker update  --cpu-shares 512  --cpuset-cpus=0,3 --memory 512m ubuntu
```

4.5.2.22 wait

Syntax: **docker wait** *container* **[***container...***]**

Function: Waits for a container to stop and print the exit code of the container:

Parameter description: none.

Example:

Run the following command to start a container named **busybox**:

```
$ sudo docker start -i -a busybox
```

Run the **docker wait** command:

```
$ sudo docker wait busybox
0
```

Wait until the busybox container exits. After the busybox container exits, the exit code **0** is displayed.

4.5.3 Image Management

4.5.3.1 build

Syntax: **docker build [***options***]** *path* | *URL* | **-**

Function: Builds an image using the Dockerfile in the specified path.

Parameter description: Common parameters are as follows. For details about more parameters, see the **docker help build** command section.

Table 4-4 Parameter description

Parameter	Description
--force-rm=false	Deletes containers generated during the build process even if the build is not successful.
--no-cache=false	Builds cache without using cache.
-q, --quiet=false	Prevents the redundant information generation during the build.
--rm=true	Deletes the container generated during the build after the build is successful.
-t, --tag=""	Tag name of the image generated during the build.
--build-arg=[]	Configures the build parameters.
--label=[]	Image-related parameters. The description of each parameter is similar to that of the **create** command.
--isolation	Container isolation method.
--pull	Obtains the latest image during the build.

Dockerfile Command

Dockerfile is used to describe how to build an image and automatically build a container. The format of all **Dockerfile** commands is *instruction arguments*.

FROM Command

Syntax: **FROM** *image* or **FROM** *image*:*tag*

Function: Specifies a basic image, which is the first command for all Dockerfile files. If the tag of a basic image is not specified, the default tag name **latest** is used.

RUN Command

Syntax: **RUN** *command* (for example, **run in a shell - `/bin/sh -c`**) or

RUN [*executable*, *param1*, *param2* ... **]** (in the **exec** command format)

Function: Runs any command in the image specified by the **FROM** command and then commits the result. The committed image can be used in later commands. The **RUN** command is equivalent to:

docker run *image command*

docker commit *container_id*

Remarks

The number sign (#) is used to comment out.

MAINTAINER Command

Syntax: **MAINTAINER** *name*

Function: Specifies the name and contact information of the maintenance personnel.

ENTRYPOINT Command

Syntax: **ENTRYPOINT cmd** *param1 param2...* or **ENTRYPOINT [***"cmd", "param1", "param2"...***]**

Function: Configures the command to be executed during container startup.

USER Command

Syntax: **USER** *name*

Function: Specifies the running user of memcached.

EXPOSE Command

Syntax: **EXPOSE** *port* **[***port***...]**

Function: Enables one or more ports for images.

ENV Command

Syntax: **ENV** *key value*

Function: Configures environment variables. After the environment variables are configured, the **RUN** commands can be subsequently used.

ADD Command

Syntax: **ADD** *src dst*

Function: Copies a file from the *src* directory to the *dest* directory of a container. *src* indicates the relative path of the source directory to be built. It can be the path of a file or directory, or a remote file URL. *dest* indicates the absolute path of the container.

VOLUME Command

Syntax: **VOLUME ["***mountpoint***"]**

Function: Creates a mount point for sharing a directory.

WORKDIR Command

Syntax: **workdir** *path*

Function: Runs the **RUN**, **CMD**, and **ENTRYPOINT** commands to set the current working path. The current working path can be set multiple times. If the current working path is a relative path, it is relative to the previous **WORKDIR** command.

CMD command

Syntax: **CMD [***"executable","param1","param2"***]** (This command is similar to the **exec** command and is preferred.)

CMD ["*param1***","***param2***"]** (The parameters are the default parameters for ENTRYPOINT.)

CMD *command param1 param2* (This command is similar to the **shell** command.)

Function: A Dockerfile can contain only one CMD command. If there are multiple CMD commands, only the last one takes effect.

ONBUILD Commands

Syntax: **ONBUILD [***other commands***]**

Function: This command is followed by other commands, such as the **RUN** and **COPY** commands. This command is not executed during image build and is executed only when the current image is used as the basic image to build the next-level image.

The following is a complete example of the Dockerfile command that builds an image with the sshd service installed.

```
FROM busybox
ENV  http proxy http://192.168.0.226:3128
ENV  https proxy https://192.168.0.226:3128
RUN apt-get update && apt-get install -y openssh-server
RUN mkdir -p /var/run/sshd
```

```
EXPOSE 22
ENTRYPOINT /usr/sbin/sshd -D
```

Example:

1. Run the following command to build an image using the preceding Dockerfile:

   ```
   $ sudo docker build -t busybox:latest
   ```

2. Run the following command to view the generated image:

   ```
   docker images | grep busybox
   ```

4.5.3.2 history

Syntax: **docker history [***options***]** *image*

Function: Displays the change history of an image.

Parameter description:

-H, --human=true

--no-trunc=false: Does not delete any output.

-q and **--quiet=false**: Display only IDs.

Example:

```
$ sudo docker history busybox:test
IMAGE             CREATED           CREATED BY        SIZE              COMMENT
be4672959e8b      15 minutes ago    bash              23B
21970dfada48      4 weeks ago                         128MB             Imported from
-
```

4.5.3.3 images

Syntax: **docker images [***options***] [***name***]**

Function: Lists existing images. The intermediate image is not displayed if no parameter is configured.

Parameter description:

-a and **--all=false**: Display all images.

-f and **--filter=[]**: Specify a filtering value, for example, **dangling=true**.

--no-trunc=false: Does not delete any output.

-q and **--quiet=false**: Display only IDs.

Example:

```
$ sudo docker images
REPOSITORY        TAG               IMAGE ID          CREATED           SIZE
busybox           latest            e02e811dd08f      2 years ago       1.09MB
```

4.5.3.4 import

Syntax: **docker import URL|- [***repository***[***:tag***]]**

Function: Imports a .tar package that contains rootfs as an image. This parameter corresponds to the **docker export** command.

Parameter description: none.

Example:

Run the following command to generate a new image for **busybox.tar** exported using the **docker export** command:

```
$ sudo docker import busybox.tar busybox:test
sha256:a79d8ae1240388fd3f6c49697733c8bac4d87283920defc51fb0fe4469e30a4f
$ sudo docker images
REPOSITORY         TAG              IMAGE ID          CREATED           SIZE
busybox            test             a79d8ae12403      2 seconds ago      1.3MB
```

4.5.3.5 load

Syntax: **docker load [***options***]**

Function: Reloads an image from .tar package obtained by running the **docker save** command. This parameter corresponds to the **docker save** command.

Parameter description:

-i and **--input=""** can be used.

Example:

```
$ sudo docker load -i busybox.tar
Loaded image ID:
sha256:e02e811dd08fd49e7f6032625495118e63f597eb150403d02e3238af1df240ba
$ sudo docker images
REPOSITORY         TAG              IMAGE ID          CREATED           SIZE
busybox            latest           e02e811dd08f      2 years ago        1.09MB
```

4.5.3.6 login

Syntax: **docker login [***options***] [***server***]**

Function: Logs in to an image server. If no server is specified, the system logs in to **https://index.docker.io/v1/** by default.

Parameter description:

-e and **--email=""**: Email address.

-p and **--password=""**: Password.

-u and **--username=""**: User name.

Example:

```
$ sudo docker login
```

4.5.3.7 logout

Syntax: **docker logout [***server***]**

Function: Logs out of an image server. If no server is specified, the system logs out of **https://index.docker.io/v1/** by default.

Parameter description: none.

Example:

```
$ sudo docker logout
```

4.5.3.8 pull

Syntax: **docker pull [***options***]** *name***[***:tag***]**

Function: Pulls an image from an official or private registry.

Parameter description:

-a and **--all-tags=false**: Download all images in a registry. (A registry can be tagged with multiple tags. For example, a busybox registry may have multiple tags, such as **busybox:14.04**, **busybox:13.10**, **busybox:latest**. If **-a** is used, all busybox images with tags are pulled.)

Example:

1. Run the following command to obtain the Nginx image from the official registry:

   ```
   $ sudo docker pull nginx
   Using default tag: latest
   latest: Pulling from official/nginx
   94ed0c431eb5: Pull complete
   9406c100a1c3: Pull complete
   aa74daafd50c: Pull complete
   Digest: sha256:788fa27763db6d69ad3444e8ba72f947df9e7e163bad7c1f5614f8fd27a311c3
   Status: Downloaded newer image for nginx:latest
   ```

   When an image is pulled, the system checks whether the dependent layer exists. If yes, the local layer is used.

2. Pull an image from a private registry.

   Run the following command to pull the Fedora image from the private registry, for example, the address of the private registry is **192.168.1.110:5000**:

   ```
   $ sudo docker pull 192.168.1.110:5000/fedora
   ```

4.5.3.9 push

Syntax: **docker push** *name***[***:tag***]**

Function: Pushes an image to the image registry.

Parameter description: none.

Example:

1. Run the following command to push an image to the private image registry at 192.168.1.110:5000.
2. Label the image to be pushed. (The **docker tag** command is described in the following section.) In this example, the image to be pushed is busybox:sshd.

```
$ sudo docker tag ubuntu:sshd 192.168.1.110:5000/busybox:sshd
```

3. Run the following command to push the tagged image to the private image registry:

```
$ sudo docker push 192.168.1.110:5000/busybox:sshd
```

During the push, the system automatically checks whether the dependent layer exists in the image registry. If yes, the layer is skipped.

4.5.3.10 rmi

Syntax: **docker rmi [***options***]** *image* **[***image...***]**

Function: Deletes one or more images. If an image has multiple tags in the image library, only the untag operation is performed when the image is deleted. If the image has only one tag, the dependent layers are deleted in sequence.

Parameter description:

-f and **--force=false**: Forcibly delete an image.

--no-prune=false: Does not delete parent images without tags.

Example:

```
$ sudo docker rmi 192.168.1.110:5000/busybox:sshd
```

4.5.3.11 save

Syntax: **docker save [***options***]** *image* **[***image...***]**

Function: Saves an image to a TAR package. The output is **STDOUT** by default.

Parameter description:

-o and **--output=""**: Save an image to a file rather than STDOUT.

Example:

```
$ sudo docker save -o nginx.tar nginx:latest
$ ls
nginx.tar
```

4.5.3.12 search

Syntax: **docker search** *options TERM*

Function: Searches for a specific image in the image registry.

Parameter description:

--automated=false: Displays the automatically built image.

--no-trunc=false: Does not delete any output.

-s and **--stars=0**: Display only images of a specified star level or higher.

Example:

1. Run the following command to search for Nginx in the official image library:

```
$ sudo docker search nginx
NAME                          DESCRIPTION                                     STARS
OFFICIAL          AUTOMATED
```

```
nginx                          Official build of Nginx.                    11873
[OK]
jwilder/nginx-proxy            Automated Nginx reverse proxy for docker con...
1645                             [OK]
richarvey/nginx-php-fpm         Container running Nginx + PHP-FPM capable of...
739                              [OK]
linuxserver/nginx              An Nginx container, brought to you by LinuxS...   74
bitnami/nginx                 Bitnami nginx Docker Image                   70
[OK]
tiangolo/nginx-rtmp            Docker image with Nginx using the nginx-rtmp...
51                               [OK]
```

2. Run the following command to search for busybox in the private image library. The address of the private image library must be added during the search.

```
$ sudo docker search 192.168.1.110:5000/busybox
```

4.5.3.13 tag

Syntax: **docker tag [***options***] *image*[*:tag*] [*registry host/*][*username/*]*name*[*:tag*]**

Function: Tags an image to a registry.

Parameter description:

-f or **--force=false**: Forcibly replaces the original image when the same tag name exists.

Example:

```
$ sudo docker tag busybox:latest busybox:test
```

4.5.4 Statistics

4.5.4.1 events

Syntax: **docker events [***options***]**

Function: Obtains real-time events from the docker daemon.

Parameter description:

--since="": Displays events generated after the specified timestamp.

--until="": Displays events generated before the specified timestamp.

Example:

After the **docker events** command is executed, a container is created and started by running the **docker run** command. create and start events are output.

```
$ sudo docker events
2019-08-28T16:23:09.338838795+08:00 container create
53450588a20800d8231aa1dc4439a734e16955387efb5f259c47737dba9e2b5e
(image=busybox:latest, name=eager wu)
2019-08-28T16:23:09.339909205+08:00 container attach
53450588a20800d8231aa1dc4439a734e16955387efb5f259c47737dba9e2b5e
(image=busybox:latest, name=eager wu)
2019-08-28T16:23:09.397717518+08:00 network connect
e2e20f52662f1ee2b01545da3b02e5ec7ff9c85adf688dce89a9eb73661dedaa
```

```
(container=53450588a20800d8231aa1dc4439a734e16955387efb5f259c47737dba9e2b5e,
name=bridge, type=bridge)
2019-08-28T16:23:09.922224724+08:00 container start
53450588a20800d8231aa1dc4439a734e16955387efb5f259c47737dba9e2b5e
(image=busybox:latest, name=eager wu)
2019-08-28T16:23:09.924121158+08:00 container resize
53450588a20800d8231aa1dc4439a734e16955387efb5f259c47737dba9e2b5e (height=48,
image=busybox:latest, name=eager_wu, width=210)
```

4.5.4.2 info

Syntax: **docker info**

Function: Displays the Docker system information, including the number of containers, number of images, image storage driver, container execution driver, kernel version, and host OS version.

Parameter description: none.

Example:

```
$ sudo docker info
Containers: 4
 Running: 3
 Paused: 0
 Stopped: 1
Images: 45
Server Version: 18.09.0
Storage Driver: devicemapper
 Pool Name: docker-thinpool
 Pool Blocksize: 524.3kB
 Base Device Size: 10.74GB
 Backing Filesystem: ext4
 Udev Sync Supported: true
 Data Space Used: 11GB
 Data Space Total: 51GB
 Data Space Available: 39.99GB
 Metadata Space Used: 5.083MB
 Metadata Space Total: 532.7MB
 Metadata Space Available: 527.6MB
 Thin Pool Minimum Free Space: 5.1GB
 Deferred Removal Enabled: true
 Deferred Deletion Enabled: true
 Deferred Deleted Device Count: 0
......
```

4.5.4.3 version

Syntax: **docker version**

Function: Displays the Docker version information, including the client version, server version, Go version, and OS and Arch information.

Parameter description: none.

Example:

```
$ sudo docker version
Client:
 Version:         18.09.0
 EulerVersion:     18.09.0.48
 API version:      1.39
 Go version:       go1.11
 Git commit:       cbf6283
 Built:           Mon Apr  1 00:00:00 2019
 OS/Arch:         linux/arm64
 Experimental:     false

Server:
 Engine:
  Version:        18.09.0
  EulerVersion:    18.09.0.48
  API version:     1.39 (minimum version 1.12)
  Go version:      go1.11
  Git commit:      cbf6283
  Built:          Mon Apr  1 00:00:00 2019
  OS/Arch:        linux/arm64
  Experimental:    false
```