

Huawei openEuler Certification Training

# HCIP-openEuler

## Lab Guide

ISSUE: 1.0



HUAWEI TECHNOLOGIES CO., LTD.

**Copyright © Huawei Technologies Co., Ltd. 2024. All rights reserved.**

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

### **Trademarks and Permissions**



HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

### **Notice**

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address:      Huawei Industrial Base Bantian, Longgang Shenzhen 518129 People's Republic of China

Website:      <https://e.huawei.com>

## Huawei Certification System

Huawei Certification is an integral part of the company's Platform + Ecosystem strategy. It supports the development of ICT infrastructure that features Cloud-Pipe-Device synergy. Our certification is always evolving to reflect the latest trends in ICT development. Huawei Certification consists of three categories: ICT Infrastructure Certification, Basic Software & Hardware Certification, and Cloud Platform & Services Certification, making it the most extensive technical certification program in the industry.

Huawei offers three levels of certification: Huawei Certified ICT Associate (HCIA), Huawei Certified ICT Professional (HCIP), and Huawei Certified ICT Expert (HCIE).

Our programs cover all ICT fields and follow the industry's trend of ICT convergence. With our leading talent development system and certification standards, we are committed to fostering new digital ICT talent and building a sound ICT talent ecosystem.

HCIP-openEuler is mainly for frontline engineers from Huawei and representative offices and readers who wish to learn openEuler O&M technologies. HCIP-openEuler certification covers common openEuler enterprise service management, openEuler HA cluster architecture, openEuler storage management, openEuler automated O&M, Linux shell scripts, openEuler system security hardening, and openEuler system monitoring.

Huawei certification helps you unlock opportunities to advance your career and take one more step towards the top of the industry.

## Huawei Career Certification

ICT Infrastructure	Datacom Domain		Optical Domain		Wireless Domain		Cloud Platform & Services		Basic Software & Hardware		IT Infrastructure Domain		Huawei Certified ICT Expert				
	5G	Transmission	5G	WLAN	5G	LTE	Cloud Computing	Big Data	GaussDB	IoT	openEuler	openGauss	HarmonyOS	AI	Kunpeng	HCIE	HCIP
Cloud Platform & Services	Datacom	Optical	Wireless	WLAN	Cloud Computing	Cloud Service	Big Data	GaussDB	IoT	openEuler	openGauss	HarmonyOS	AI	Kunpeng	Huawei Certified ICT Expert	Huawei Certified ICT Expert	Huawei Certified ICT Associate
Basic Software & Hardware	5G	Transmission	5G	WLAN	Cloud Computing	Cloud Service	Big Data	GaussDB	IoT	openEuler	openGauss	HarmonyOS	AI	Kunpeng	Huawei Certified ICT Expert	Huawei Certified ICT Expert	Huawei Certified ICT Associate
IT Infrastructure Domain	Datacom	Optical	Wireless	WLAN	Cloud Computing	Cloud Service	Big Data	GaussDB	IoT	openEuler	openGauss	HarmonyOS	AI	Kunpeng	Huawei Certified ICT Professional	Huawei Certified ICT Professional	Huawei Certified ICT Associate
Datacom Domain	5G	Transmission	5G	WLAN	Cloud Computing	Cloud Service	Big Data	GaussDB	IoT	openEuler	openGauss	HarmonyOS	AI	Kunpeng	Huawei Certified ICT Professional	Huawei Certified ICT Professional	Huawei Certified ICT Associate
Optical Domain	5G	Transmission	5G	WLAN	Cloud Computing	Cloud Service	Big Data	GaussDB	IoT	openEuler	openGauss	HarmonyOS	AI	Kunpeng	Huawei Certified ICT Professional	Huawei Certified ICT Professional	Huawei Certified ICT Associate
Wireless Domain	5G	Transmission	5G	WLAN	Cloud Computing	Cloud Service	Big Data	GaussDB	IoT	openEuler	openGauss	HarmonyOS	AI	Kunpeng	Huawei Certified ICT Professional	Huawei Certified ICT Professional	Huawei Certified ICT Associate

# About This Document

---

## Overview

This document is an HCIP-openEuler certification training course and is intended for trainees who are going to take the HCIP-openEuler exam or readers who want to learn how to build enterprise services, master shell scripts, or perform automated O&M using Zabbix or Salt on openEuler and other Linux distributions.

## Description

This lab guide describes five labs about how to set up Apache, NGINX, DNS, and MySQL servers based on openEuler and how to perform LAMP-related operations through a comprehensive lab.

- Lab 1: Apache lab. This lab involves service installation and basic management, helping you further understand Apache.
- Lab 2: NGINX lab. This lab involves NGINX installation, static resource access, and reverse proxy.
- Lab 3: DNS lab. This lab combines lab 2 to configure forward and reverse DNS resolution for the NGINX server.
- Lab 4: MySQL lab. This lab involves how to install and perform basic operations on MySQL, including adding, deleting, querying, and modifying MySQL.
- Lab 5: Comprehensive lab. This lab describes how to set up WordPress, a LAMP typical application, to help you further understand enterprise services.

## Background Knowledge Required

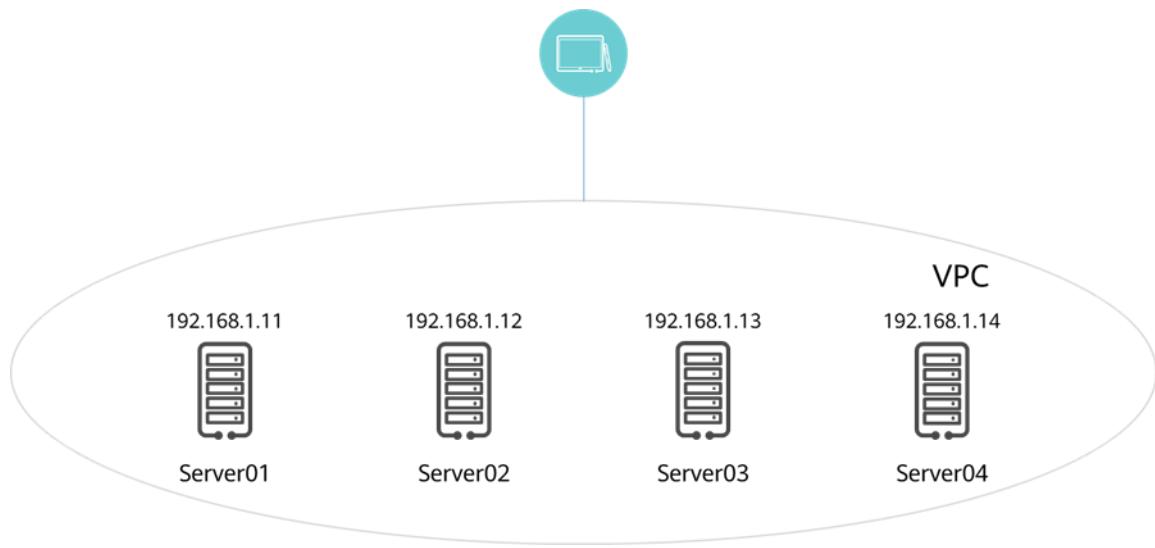
This course is for Huawei's basic certification. To better understand this course, you need to:

- Have basic Linux knowledge. You are advised to complete HCIA-openEuler learning and pass the HCIA-openEuler certification exam.

## Lab Environment Overview

Four Elastic Cloud Servers (ECSs) are required: **Server01** to **Server04** with IP addresses ranging from 192.168.1.11 to 192.168.1.14. **Server01** is for Apache, **Server02** for NGINX, **Server03** for DNS, and **Server04** for MySQL. To view the lab result on the browser, bind elastic IP addresses (EIPs) to the ECSs. Because the EIPs are not manually specified, use the actual EIPs in your labs.

In this environment, the EIPs of **Server01**, **Server02**, **Server03**, and **Server04** are 124.70.153.72, 123.60.63.199, 121.37.184.67, and 124.70.167.64, respectively. See the following figure.



For the ECS security group, allow both inbound and outbound traffic, as shown in the following figure.

Priority	Action	Type	Protocol & Port	Source
<input type="checkbox"/> 1	Allow	IPv4	All	0.0.0.0/0 
<input type="checkbox"/> 100	Allow	IPv6	All	default 

## Lab Environment Preparation

### Checking Devices

Before starting the labs, each group of trainees should apply for ECSs on Huawei Cloud according to the following table.

ECS Name	Specifications	Remarks
Server01	2 vCPUs   4 GiB   s7.large.2	
Server02	2 vCPU   4 GiB   s7.large.2	
Server03	1 vCPU   1 GiB   s7.small.1	
Server04	2 vCPUs   4 GiB   s7.large.2	

# Contents

---

<b>About This Document .....</b>	<b>3</b>
Overview .....	3
Description .....	3
Background Knowledge Required .....	3
Lab Environment Overview.....	3
<b>1 Apache Configurations.....</b>	<b>1</b>
1.1 Installing and Testing Apache .....	1
1.1.1 Procedure.....	1
1.1.2 Quiz.....	3
1.2 Configuring the Apache Home Page .....	3
1.2.1 Procedure.....	3
1.2.2 Quiz.....	7
1.3 Loading and Unloading an Apache DSO Module .....	7
1.3.1 Configuration Roadmap .....	7
1.3.2 Procedure.....	7
1.4 Changing the Working Mode of the MPM .....	9
1.4.1 Procedure.....	9
1.4.2 Quiz.....	10
1.5 Configuring Apache Persistent Connection .....	11
1.5.1 Procedure.....	11
1.5.2 Quiz.....	12
1.6 Configuring Apache Static Resources .....	12
1.6.1 Specifying Static Resources by File System Path.....	12
1.6.2 Setting Access Permissions for Static Resources (Based on Source Addresses) .....	15
1.6.3 Setting Access Permissions for Static Resources (Based on Accounts) .....	17
1.6.4 Quiz.....	18
1.7 Configuring Apache Virtual Hosts.....	18
1.8 Configuring the HTTPS Service .....	20
<b>2 NGINX Configurations.....</b>	<b>25</b>
2.1 Installing and Testing NGINX .....	25
2.1.1 Preparing Resources.....	25
2.1.2 Procedure.....	25
2.2 Performing Basic NGINX Configurations .....	26
2.2.1 Configuring Static Resource Access.....	26

2.2.2 Configuring Virtual Hosts .....	29
2.2.3 Configuring the Location Directive.....	30
2.3 Configuring Reverse Proxies and Load Balancing with NGINX .....	34
2.3.1 Configuring Reverse Proxies with NGINX .....	34
2.3.2 Configuring Load Balancing with NGINX.....	36
<b>3 DNS Configurations .....</b>	<b>41</b>
3.1 Introduction.....	41
3.1.1 About This Lab.....	41
3.2 Installing the DNS Software.....	41
3.2.1 Preparing Resources.....	41
3.2.2 Installing the DNS Software .....	41
3.3 Setting Up the Master DNS Server .....	42
3.4 Configuring Reverse DNS Resolution .....	44
<b>4 MySQL Configurations .....</b>	<b>46</b>
4.1 Installing and Initializing MySQL.....	46
4.1.1 Preparing Resources.....	46
4.1.2 Installing MySQL.....	46
4.1.3 Logging In to and Initializing MySQL.....	48
4.1.4 Using mysqladmin .....	49
4.2 Performing Comprehensive MySQL Practices.....	51
<b>5 LAMP Practices.....</b>	<b>60</b>
5.1 Introduction .....	60
5.2 Interconnecting Components in the Early Stage .....	60
5.2.1 Interconnecting Apache with PHP .....	60
5.2.2 Interconnecting PHP with MySQL .....	62
5.3 Performing LAMP Practices .....	64
5.3.1 Introduction .....	64
5.3.2 Preparing Resources.....	64
5.3.3 Installing and Testing WordPress .....	65

# 1

# Apache Configurations

## 1.1 Installing and Testing Apache

### 1.1.1 Procedure

Step 1 Purchase **Server01** based on the lab environment description and networking.

Step 2 Run the **dnf** command to install httpd.

Log in to the installed openEuler, run the **dnf install -y httpd** command, and wait until the installation is complete.

```
[root@apache-http mnt]# dnf install -y httpd
```

```
[root@apache-http mnt]# dnf install -y httpd
local
Last metadata expiration check: 0:00:02 ago on [REDACTED] AM CST.
Dependencies resolved.
=====
 Package                               Architecture
=====
Installing:
 httpd                                x86_64
Installing dependencies:
 apr                                  x86_64
 apr-util                             x86_64
 httpd-filesystem                     noarch
 httpd-tools                          x86_64
 mailcap                             noarch
 mariadb-connector-c                 x86_64
 mod_http2                           x86_64
 openEuler-logos                      noarch
Transaction Summary
=====
Install 9 Packages
```

Step 3 Start httpd and test it.

Run the **httpd -v** command to view the version.

```
[root@apache-http ~]# httpd -v
Server version: Apache/2.4.51 (Unix)
Server built: Mar 18 2022 00:00:00
```

```
[root@apache-http ~]# httpd -v
Server version: Apache/2.4.51 (Unix)
Server built: Mar 18 2022 00:00:00
```

Run the **httpd -t** command to check whether the configuration file is correct.

```
[root@apache-http ~]# httpd -t
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using
fe80::f21:38fa:dc6d:d4e9%ens33. Set the 'ServerName' directive globally to suppress this message
Syntax OK
```

```
[root@apache-http ~]# httpd -t
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using fe80::f21:38fa:dc6d:d4e9%ens33. Set the 'ServerName' directive globally to suppress this message
Syntax OK
```

Modify the **ServerName** configuration in line 97 of the **/etc/httpd/conf/httpd.conf** configuration file to suppress the error message.

```
94 #
95 # If your host doesn't have a registered DNS
96 #
97 ServerName www.test.com:80
98
99
```

Save the modification and exit. Check whether the configuration file is correct.

```
[root@apache-http ~]# httpd -t
Syntax OK
```

Run the **systemctl enable httpd --now** command to start the httpd service and set it to start upon system startup. Then run the **systemctl status httpd** command to check the httpd service status.

```
[root@apache-http ~]# systemctl enable httpd --now
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service →
/usr/lib/systemd/system/httpd.service.
[root@apache-http ~]# systemctl status httpd
```

```
[root@apache-http ~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
  Active: active (running) since Sat 2023-06-10 11:28:58 CST; 44s ago
    Docs: man:httpd.service(8)
 Main PID: 1564 (httpd)
   Status: "Total requests: 0; Idle/Busy workers 100/0;Requests/sec: 0; Bytes served/sec: 0 B/sec"
      Tasks: 177 (limit: 21420)
     Memory: 21.3M
        CPU: 0.000 CPU(s)
       CGroup: /system.slice/httpd.service
               └─1564 /usr/sbin/httpd -DFOREGROUND
                  ├─1565 /usr/sbin/httpd -DFOREGROUND
                  ├─1566 /usr/sbin/httpd -DFOREGROUND
                  ├─1567 /usr/sbin/httpd -DFOREGROUND
                  ├─1568 /usr/sbin/httpd -DFOREGROUND

Jun 10 11:28:58 apache-http systemd[1]: Starting The Apache HTTP Server...
Jun 10 11:28:58 apache-http systemd[1]: Started The Apache HTTP Server.
[root@apache-http ~]#
```

Note: If the server fails to be started, restart it and try again.

**Step 4** Alternatively, use the following methods for testing the httpd service status:

Method 1:

Run the **ss -lnp | grep 80** command to check whether port 80 is in the listening state.

```
[root@apache-http ~]# ss -lnp | grep 80
```

```
[root@apache-http ~]# ss -lnp | grep 80
u_str LISTEN 0      100          /run/httpd/cgisock.1564 24803
tcp    LISTEN 0      511          *:80
```

Method 2:

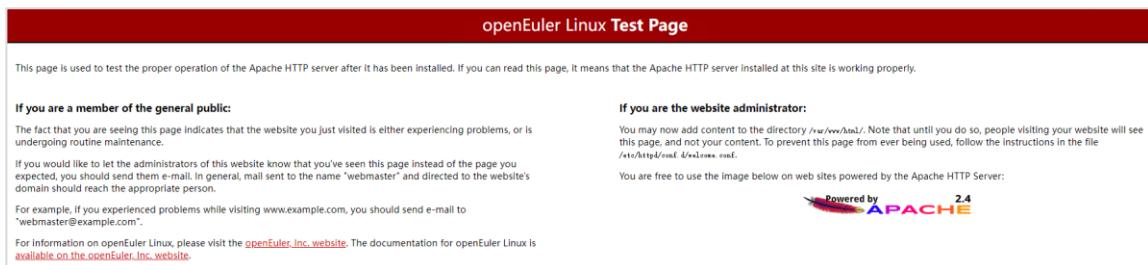
Run the **curl** command on the HTTP server to access port 80.

```
[root@apache-http ~]# curl 127.0.0.1:80
```

```
[root@apache-http ~]# curl 127.0.0.1:80
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Test Page for the Apache HTTP Server on openEuler Linux</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <style type="text/css">
      /*<![CDATA[*/
      body {
        background-color: #fff;
        color: #000;
        font-size: 0.9em;
        font-family: sans-serif, helvetica;
        margin: 0;
        padding: 0;
      }
      :link {
        color: #c00;
      }
    <![CDATA]]>
```

Method 3:

Enter the EIP of the HTTP server in the address box of the browser and check whether the test page is displayed.



## 1.1.2 Quiz

Check whether the test page can be accessed using method 3. If not, why?

The access may fail. If both methods 1 and 2 succeeded but method 3 failed, there is a high probability that port 80 is not enabled on the HTTP server. In this case, check the firewall or security group.

## 1.2 Configuring the Apache Home Page

### 1.2.1 Procedure

Step 1 Configure the listening IP address and port of the server.

Requirement: Clients are allowed to access the HTTP server only through port 81 at the server IP address (192.168.1.11).

Comment out the listening IP address and port number in the main configuration file, as shown in the following figure.

```
# Change this to Listen on a specific IP address and port
# httpd.service is enabled to run at boot time
# available when the service starts. See /usr/share/doc/httpd/html
#
#Listen 12.34.56.78:80
#Listen 80
```

Create a configuration file **port.conf** in **/etc/httpd/conf.d** and add the following content to the file:

```
Listen 192.168.1.11:81
```

Run the following command to reload the httpd configuration file:

```
[root@apache-http conf.d]# systemctl reload httpd
```

Run the **curl 192.168.1.11:81**, **curl 192.168.1.11:80**, and **curl 127.0.0.1:81** commands to check whether the test page can be accessed.

```
[root@apache-http ~]# curl 127.0.0.1:81
curl: (7) Failed to connect to 127.0.0.1 port 81 after 0 ms: Connection refused
[root@apache-http ~]# curl 192.168.1.11:80
curl: (7) Failed to connect to 192.168.1.11 port 80 after 0 ms: Connection refused
[root@apache-http ~]# curl 192.168.1.11:81
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
    <head>
        <title>Test Page for the Apache HTTP Server on openEuler Linux</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <style type="text/css">
            /*<![CDATA[*/
            body {
                background-color: #fff;
                color: #000;
```

- Question: If the content in the main configuration file is not commented out, can the effect in the requirement be achieved?

Answer: No. If you do not comment out the content, clients can access the service through port 80.

- Question: Can I directly modify the main configuration file to achieve the effect specified in the requirement?

Answer: Yes, you can. However, if the main configuration file is modified, it is difficult to maintain the file. You are advised to add the mentioned configuration file to the **conf.d** directory.

## Step 2 Configure the home page.

Requirement: "hello, openEuler" is returned after the HTTP server is accessed.

Create an **Index.html** file in the **/var/www/html** directory and add "**hello, openEuler**" to the file, as shown in the following figure.

```
[root@apache-http html]# rm -rf index.html
[root@apache-http html]# touch index.html
[root@apache-http html]# echo "hello, openEuler" > index.html
[root@apache-http html]# cat index.html
hello, openEuler
```

Run the **systemctl reload httpd** command and access the server again, as shown in the following figure.

```
[root@apache-http ~]# systemctl reload httpd
[root@apache-http ~]# curl 192.168.1.11:81
hello, openEuler
[root@apache-http ~]#
```

### Step 3 Change the directory for storing the home page.

Requirement: Create the **/home/source** directory and move **index.html** to the directory to enable the server to return "**hello, openEuler**".

Comment out the configuration of the directory for storing static resources in the main configuration file, as shown in the following figure.

```
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
#DocumentRoot "/var/www/html"
```

Create a **home/source** directory and cut the **index.html** file to the directory, as shown in the following figure.

```
[root@apache-http html]# mkdir /home/source
[root@apache-http html]# mv index.html /home/source/
```

Create a configuration file **source.conf** in **conf.d** and enter the following content:

```
DocumentRoot "/home/source"
```

See the following figure.

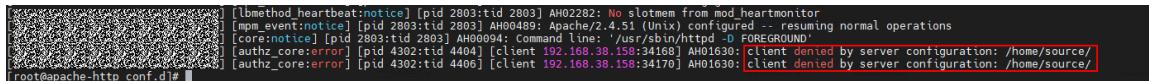
```
[root@apache-http conf.d]# touch source.conf
[root@apache-http conf.d]# echo 'DocumentRoot "/home/source"' > source.conf
```

Run the **systemctl reload httpd** command and access the server again, as shown in the following figure.

```
[root@apache-http conf.d]# curl 192.168.1.11:81
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Test Page for the Apache HTTP Server on openEuler Linux</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <style type="text/css">
      /*!<![CDATA[*/
      body {
        background-color: #fff;
        color: #000;
        font-size: 0.9em;
        font-family: sans-serif,helvetica;
        margin: 0;
        padding: 0;
      }
      :link {
        color: #c00;
      }
    </style>
  </head>
  <body>
    <h1>Apache Test Page</h1>
    <p>This page is generated by the Apache HTTP Server. It is running on an openEuler Linux system. The server is configured to serve static files from the /home/source directory. If you see this page, it means your configuration is correct. You can now proceed to test other features of the server, such as dynamic content generation or error handling. If you encounter any issues, please refer to the Apache documentation or seek help from the community forums. Happy coding!</p>
  </body>
</html>
```

- Question: Why is the expected content not returned?

Check the httpd log file. It is found that the access to the **/home/source** directory is denied, as shown in the following figure.



```
[root@apache-http conf.d] [lbbmethod_heartbeat:notice] [pid 2803:tid 2803] AH02282: No slotmem from mod_heartmonitor
[mpm_event:notice] [pid 2803:tid 2803] AH00489: Apache/2.4.51 (Unix) configured -- resuming normal operations
[core:notice] [pid 2803:tid 2803] AH00094: Command line: '/usr/sbin/httpd -D FOREGROUND'
[authz_core:error] [pid 4302:tid 4404] [client 192.168.38.158:34168] AH01630: Client denied by server configuration: /home/source/
[authz_core:error] [pid 4302:tid 4406] [client 192.168.38.158:34170] AH01630: Client denied by server configuration: /home/source/
```

Add the following content to the **source.conf** file to grant the access permission on the directory:

```
<Directory "/home/source">
  AllowOverride None
  # Allow open access:
  Require all granted
</Directory>
```

See the following figure.

```
[root@apache-http conf.d]# cat source.conf
DocumentRoot "/home/source"
<Directory "/home/source">
  AllowOverride None
  # Allow open access:
  Require all granted
</Directory>
```

Run the **systemctl reload httpd** command and access the server again. The result meets the requirement, as shown in the following figure.

```
[root@apache-http conf.d]# systemctl reload httpd
[root@apache-http conf.d]# curl 192.168.1.11:81
hello, openEuler
```

#### Step 4 Change the name of the home page.

Change **index.html** to **Index.html** and access the service again. The system returns to the test page, as shown in the following figure.

```
[root@apache-http source]# mv index.html Index.html
[root@apache-http source]# systemctl reload httpd
[root@apache-http source]# curl 192.168.38.158:81
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Test Page for the Apache HTTP Server on openEuler Linux</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <style type="text/css">
      /*<![CDATA[*/
      body {
        background-color: #fff;
        color: #000;
        font-size: 0.9em;
        margin: 0;
        padding: 0;
      }
    </style>
  </head>
  <body>
```

Add the following content to the **source.conf** file to change the static page name to **Index.html**:

```
DirectoryIndex Index.html
```

Reload the configurations and try to access the service again, as shown in the following figure.

```
[root@apache-http source]# echo "DirectoryIndex Index.html" >> /etc/httpd/conf.d/source.conf
[root@apache-http source]# cat /etc/httpd/conf.d/source.conf
DocumentRoot "/home/source"
<Directory "/home/source">
  AllowOverride None
  # Allow open access:
  Require all granted
</Directory>
DirectoryIndex Index.html
[root@apache-http source]# systemctl reload httpd
[root@apache-http source]# curl 192.168.38.158:81
hello, openEuler
```

## 1.2.2 Quiz

Can an HTTP server return different static pages for different users?

Yes, it can. This is the virtual host function of the Apache httpd service, which will be introduced and implemented later.

# 1.3 Loading and Unloading an Apache DSO Module

## 1.3.1 Configuration Roadmap

The **mod\_status** module allows an administrator to monitor the Apache running status on a web page. Use **LoadModule** directive to load the module, configure related permissions, and enable **ExtendedStatus** to enable the module.

## 1.3.2 Procedure

Step 1    Check the status of the **mod\_status** module.

Run the **httpd -M** command to check whether **mod\_status** is loaded. The command output is as follows:

```
[root@apache-http conf.modules.d]# httpd -M | grep status
```

```
status_module (shared)
```

```
[root@apache-http conf.modules.d]# httpd -M | grep status
status_module (shared)
```

## Step 2 Configure related permissions and enable **ExtendedStatus**.

Add a configuration file **module.conf** to **/etc/httpd/conf.d** and add the following content:

```
<Location /server-status>
    SetHandler server-status
    Order deny,allow
    Deny from all
    Allow from all
</Location>
ExtendedStatus On
```

## Step 3 Check the Apache status.

Reload the Apache configurations, and enter **E/P:81/server-status** in the address box of the browser to check the Apache status.

### Apache Server Status for 124.70.153.72 (via 192.168.1.11)

Server Version: Apache/2.4.51 (Unix)  
Server MPM: event  
Server Built: [REDACTED]

---

Current Time: Tuesday, [REDACTED] UTC  
Restart Time: Tuesday, [REDACTED] UTC  
Parent Server Config. Generation: 7  
Parent Server MPM Generation: 6  
Server uptime: 19 minutes 23 seconds  
Server load: 0.00 0.00 0.00  
Total accesses: 8 - Total Traffic: 14 kB - Total Duration: 6  
CPU Usage: u.05 s.08 cu.23 cs.32 - .0585% CPU load  
.00688 requests/sec - 12 B/second - 1792 B/request - .75 ms/request  
1 requests currently being processed, 74 idle workers

Slot	PID	Stopping	Connections	Threads	Async connections				
			total	accepting	busy	idle	writing	keep-alive	closing
0	2129	no	0	yes	0	25	0	0	0
1	2130	no	0	yes	0	25	0	0	0
2	2131	no	0	yes	1	24	0	0	0
Sum	3		0		1	74	0	0	0

---

.....  
.....  
.....  
.....  
.....

## Step 4 Unload the mod\_status module.

In the **/etc/httpd/conf.modules.d** directory, edit the **00-base.conf** file to comment out the mod\_status configuration as follows.

```
LoadModule socache_memcache_module modules/mod_socache_memcache.  
LoadModule socache_redis_module modules/mod_socache_redis.so  
LoadModule socache_shmcb module modules/mod_socache_shmcb.so  
# LoadModule status_module modules/mod_status.so  
LoadModule substitute_module modules/mod_substitute.so  
LoadModule suexec_module modules/mod_suexec.so  
LoadModule unique_id module modules/mod_unique_id.so
```

Reload the Apache configurations, and use `E/P:81/server-status` to access the Apache again. The following information is displayed.

## Not Found

The requested URL was not found on this server.

Step 5 Restore the module.

This module will be used in subsequent labs. Uncomment the line in step 4 and reload the Apache configurations to restore the module.

## 1.4 Changing the Working Mode of the MPM

### 1.4.1 Procedure

Step 1 Check the current working mode of the MPM.

Method 1: Run the `httpd -M` command, as shown in the following figure.

```
[root@apache-http conf.modules.d]# httpd -M | grep mpm  
mpm_event_module (shared)
```

Method 2: Use the mod\_status module, as shown in the following figure.

**Apache Server Status for 124.70.153.72 (via 192.168.1.11)**

```
Server Version: Apache/2.4.51 (Unix)  
Server MPM: event  
Server Built: Mar 9 2023 00:00:00
```

Method 3: View the related configuration file, as shown in the following figure.

```
#LoadModule mpm_prefork_module modules/mod_mpm_prefork.so  
  
# worker MPM: Multi-Processing Module implementing a hybrid  
# multi-threaded multi-process web server  
# See: http://httpd.apache.org/docs/2.4/mod/worker.html  
#  
#LoadModule mpm_worker_module modules/mod_mpm_worker.so  
  
# event MPM: A variant of the worker MPM with the goal of consuming  
# threads only for connections with active processing  
# See: http://httpd.apache.org/docs/2.4/mod/event.html  
#  
LoadModule mpm_event_module modules/mod_mpm_event.so  
~
```

### Step 2 Change the MPM working mode.

Comment out the configuration related to the event mode in the configuration file and uncomment the configuration related to the worker mode, as shown in the following figure.

```
# worker MPM: Multi-Processing Module implementing a hybrid  
# multi-threaded multi-process web server  
# See: http://httpd.apache.org/docs/2.4/mod/worker.html  
#  
LoadModule mpm_worker_module modules/mod_mpm_worker.so  
  
# event MPM: A variant of the worker MPM with the goal of consuming  
# threads only for connections with active processing  
# See: http://httpd.apache.org/docs/2.4/mod/event.html  
#  
#LoadModule mpm_event_module modules/mod_mpm_event.so  
~
```

### Step 3 Verify the configurations.

Restart the httpd service and use method 2 in step 1 to check whether the configuration takes effect, as shown in the following figure.

```
■  
Server Version: Apache/2.4.51 (Unix)  
Server MPM: worker  
Server Built: Mar 18 2022 00:00:00
```

## 1.4.2 Quiz

- After the MPM working mode is changed, can the modification take effect after the Apache is reloaded? Why?

No, it cannot. You need to restart the service for the modification to take effect. The MPM determines the working mode of the httpd processes and threads. You need to restart the service to enable the systemd to create the corresponding processes.

- Can I enable multiple working modes at the same time?

No, you cannot. If multiple working modes are enabled, the system displays the following error message.

```
[root@apache-htt conf.modules.d]# httpd -t
[Sun Jan 29 10:52:23.091632 2023] [so:warn] [pid 3089:tid 3089] AH01574: module cgid_module is already loaded, skipping
AH00534: httpd: Configuration error: More than one MPM loaded.
[root@apache-htt conf.modules.d]
```

## 1.5 Configuring Apache Persistent Connection

### 1.5.1 Procedure

Step 1 Check the status of non-persistent connections.

On the mod\_status page, the value of **keep-alive** is **0**, as shown in the following figure.

Slot	PID	Stopping	Connections		Threads		Async connections	
			total	accepting	busy	idle	writing	keep-alive
0	3676	no	0	yes	0	25	0	0
1	3677	no	0	yes	0	25	0	0
2	3678	no	0	yes	0	25	0	0
3	3854	no	0	yes	1	24	0	0
Sum	4		0		1	99	0	0

.....

.....

.....

.....

.....

Step 2 Enable persistent connection and set related parameters.

Create a **keepalived.conf** configuration file related to persistent connections in the **/etc/httpd/conf.d** directory and add the following configurations:

```
KeepAlive On
KeepAliveTimeout 20
MaxKeepAliveRequests 500
```

Step 3 Verify the configurations.

Reload the Apache configurations and check the value of **keep-alive** on the mod\_status page, as shown in the following figure.

Slot	PID	Stopping	Connections		Threads		Async connections		
			total	accepting	busy	idle	writing	keep-alive	closing
0	4421	no	1	yes	1	24	0	1	0
1	4241	no	0	yes	0	25	0	0	0
2	4242	no	0	yes	0	25	0	0	0
4	4365	no	0	yes	0	25	0	0	0
Sum	4	0	1		1	99	0	1	0

## 1.5.2 Quiz

Does a longer persistent connection indicate better effect?

No, it does not. If a connection is occupied for a long time, the number of concurrent connections increases, increasing the pressure on the server.

## 1.6 Configuring Apache Static Resources

### 1.6.1 Specifying Static Resources by File System Path

Step 1 Prepare resources.

Create a **test** directory in **/home/source**, and create files **test1** and **test2** in **test**. Enter "hello,test1" in **test1**, and soft link **test1** to **test2**.

```
[root@apache-http source]# cd /home/source
[root@apache-http source]# mkdir test
[root@apache-http source]# cd test
[root@apache-http test]# echo "hello,test1" > test1
[root@apache-http test]# ln -s test1 test2
```

```
[root@apache-http test]# ll
total 4.0K
-rw-r--r-- 1 root root 12 Jan 29 16:10 test1
lrwxrwxrwx 1 root root 5 Jan 29 16:12 test2 -> test1
[root@apache-http test]#
```

Step 2 Add configurations.

Add the following content to **/etc/httpd/conf.d/source.conf**:

```
<Directory "/test">
    Options Indexes
    AllowOverride None
    # Allow open access:
    Require all granted
</Directory>
```

```
DocumentRoot "/home/source"
<Directory "/home/source">
    Options Indexes
    AllowOverride None
    # Allow open access:
    Require all granted
</Directory>
<Directory "/test">
    Options Indexes
    AllowOverride None
    # Allow open access:
    Require all granted
</Directory>
DirectoryIndex Index.html
~
```

### Step 3 Verify the configurations.

Reload the httpd service, enter `EIP:81/test` in the address box of the browser. The following information is displayed.

## Index of /test

Name	Last modified	Size	Description
------	---------------	------	-------------

 <a href="#">Parent Directory</a>	-	-	-
 <a href="#">test1</a>			

Add the `FollowSymLinks` directive to `Options`, as shown in the following figure.

```
DocumentRoot "/home/source"
<Directory "/home/source">
    Options Indexes FollowSymLinks
    AllowOverride None
    # Allow open access:
    Require all granted
</Directory>
<Directory "/test">
    Options Indexes FollowSymLinks
    AllowOverride None
    # Allow open access:
    Require all granted
</Directory>
DirectoryIndex Index.html
~
```

Reload the httpd service and access the path again. The following information is displayed.

 <a href="#">Parent Directory</a>	-
 <a href="#">test1</a>	 12
 <a href="#">test2</a>	 12

- Question: If there is no configuration file about **/test**, can I access the corresponding resources through **192.168.1.11:81/test**?

Answer: Yes, you can.

- Question: If the **Index.html** file is created in the **test** directory, can I access the corresponding resources through **192.168.1.11:81/test**?

Answer: No, you cannot. The system returns the home page content, or indexes the current directory only when the specified home page cannot be found in the directory.

#### Step 4 Specify static resources by alias.

Add the following configuration to the **source.conf** file:

```
Alias /test2 "/home/source/test"
```

See the following figure.

```
DocumentRoot "/home/source"
<Directory "/home/source">
    Options Indexes FollowSymLinks
    AllowOverride None
    # Allow open access:
    Require all granted
</Directory>
Alias /test2 "/home/source/test"
<Directory "/home/source/test">
    Options Indexes FollowSymLinks
    AllowOverride None
    # Allow open access:
    Require all granted
</Directory>
DirectoryIndex Index.html
~
```

Reload the httpd service, enter **E/P:81/test2** in the address box of the browser. The following information is displayed.

<h2>Index of /test2</h2>			
Name	Last modified	Size	Description
 <a href="#">Parent Directory</a>	-		
 <a href="#">test1</a>	 12		
 <a href="#">test2</a>	 12		

## 1.6.2 Setting Access Permissions for Static Resources (Based on Source Addresses)

Requirement: Allow all source addresses to access **192.168.1.11:81/test2/**, except **192.168.38.1** (the IP address of the test machine after NAT, which can be determined by viewing HTTP logs).

Step 1 Modify the configuration file.

Add the following content to the configuration of the real directory of **test2** in the **source.conf** file:

```
<RequireAll>
    Require all granted
    Require not ip 192.168.38.1
</RequireAll>
```

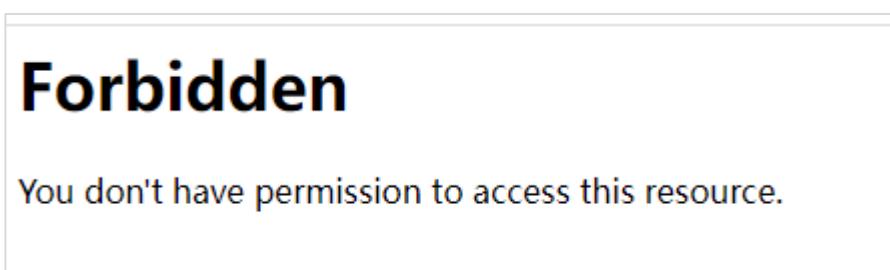
See the following figure.

```
DocumentRoot "/home/source"
<Directory "/home/source">
    Options Indexes FollowSymLinks
    AllowOverride None
    # Allow open access:
    Require all granted
</Directory>
Alias /test2 "/home/source/test"
<Directory "/home/source/test">
    Options Indexes FollowSymLinks
    <RequireAll>
        Require all granted
        Require not ip 192.168.38.1
    </RequireAll>
</Directory>

DirectoryIndex Index.html
```

Step 2 Verify the configurations.

Reload the Apache configurations and access the directory using the browser of the test machine. The following information is displayed.



Forbidden

You don't have permission to access this resource.

If you access the same directory on the server, the following information is displayed.

```
[root@apache-http source]# curl http://124.70.153.72:81/test2/
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
<head>
<title>Index of /test2</title>
</head>
<body>
<h1>Index of /test2</h1>
<table>
<tr><th valign="top"></th><td>
<description></a></td></tr>
<tr><th colspan="5"><hr></th></tr>
<tr><td valign="top"></td>
<tr><td valign="top"></td><td>
<tr><th colspan="5"><hr></th></tr>
</table>
</body></html>
[root@apache http source]#
```

View the cause of the access failure of the test machine in logs. The details are as follows.

```
[root@apache-http source]# !tail -10 /var/log/httpd/error_log
[Tue Apr 04 06:18:11.461777 2023] [lbmethod_heartbeat:notice] [pid 1002:tid 1002] AH02282: No slotmem from mod_heartmonitor
[Tue Apr 04 06:18:11.462662 2023] [mpm_event:notice] [pid 1002:tid 1002] AH00489: Apache/2.4.51 (Unix) configured -- resuming normal operations
[Tue Apr 04 06:18:23.038038 2023] [core:error] [pid 2699:tid 2723] [client 119.3.119.21:32273] AH00932: Symbolic link not allowed or link target not accessible: /home/source/test/test2
[Tue Apr 04 06:19:53.539694 2023] [mpm_event:notice] [pid 1002:tid 1002] AH00493: SIGUSR1 received. Doing graceful restart
[Tue Apr 04 06:19:53.553807 2023] [lbmethod_heartbeat:notice] [pid 1002:tid 1002] AH02282: No slotmem from mod_heartmonitor
[Tue Apr 04 06:19:53.554670 2023] [mpm_event:notice] [pid 1002:tid 1002] AH00489: Apache/2.4.51 (Unix) configured -- resuming normal operations
[Tue Apr 04 06:19:53.554670 2023] [core:notice] [pid 1002:tid 1002] AH00489: Command line: "/usr/local/apache2/bin/httpd -DFOREGROUND"
[Tue Apr 04 06:19:53.554670 2023] [core:notice] [pid 1002:tid 1002] AH00489: Parent: pid 1002
[Tue Apr 04 06:19:53.554670 2023] [core:notice] [pid 2699:tid 2991] [client 119.3.119.21:32276] AH0030: client denied by server configuration: /home/source/test/
[Tue Apr 04 06:20:36.617143 2023] [core:error] [pid 2699:tid 2991] [client 124.70.153.72:47198] AH00932: Symbolic link not allowed or link target not accessible: /home/source/test/test2
[root@apache http source]#
```

### Step 3 Use the .htaccess file to set the permissions.

Comment out the configuration added in step 1 in the **source.conf** file and add **AllowOverride All**, as shown in the following figure.

```
DocumentRoot "/home/source"
<Directory "/home/source">
    Options Indexes FollowSymLinks
    AllowOverride None
    # Allow open access:
    Require all granted
</Directory>
Alias /test2 "/home/source/test"
<Directory "/home/source/test">
    Options Indexes FollowSymLinks
    AllowOverride All
    # <RequireAll>
    # Require all granted
    # Require not ip 192.168.38.1
    # </RequireAll>
</Directory>

DirectoryIndex Index.html
~
```

Create an **.htaccess** file in **/home/source/test** and enter the following configurations:

```
<RequireAll>
    Require all granted
    Require not ip 192.168.38.1
</RequireAll>
```

See the following figure.

```
<RequireAll>
  Require all granted
  Require not ip 192.168.38.1
</RequireAll>
```

Step 4 Verify the configurations.

Refer to step 2.

### 1.6.3 Setting Access Permissions for Static Resources (Based on Accounts)

Step 1 Create an account.

Run the following command to create a **.passwd** file in the directory where the resource is located:

```
htpasswd -cb .passwd test Huawei@123
```

```
[root@apache-http test]# htpasswd -cb .passwd test Huawei@123
Adding password for user test
```

Step 2 Add configurations.

Add the following configurations to the **.htaccess** file:

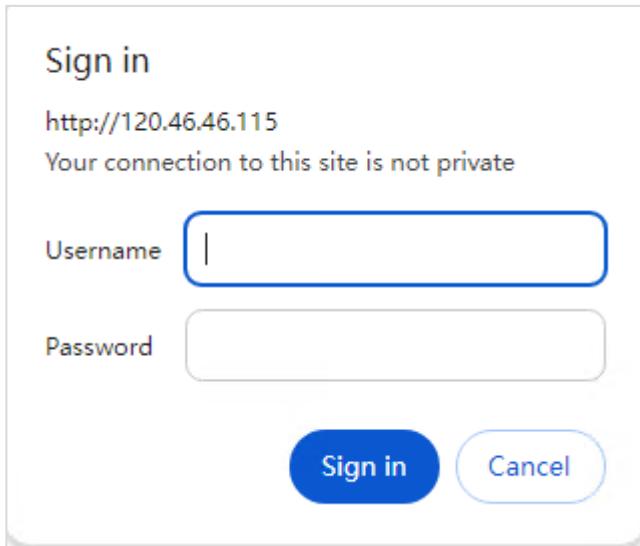
```
AuthType Basic
AuthName "http test"
AuthUserFile "/home/source/test/.passwd"
Require user test
```

See the following figure.

```
<RequireAll>
  Require all granted
  Require not ip 192.168.38.1
</RequireAll>
AuthType Basic
AuthName "http test"
AuthUserFile "/home/source/test/.passwd"
Require user test
```

Step 3 Verify the configurations.

Reload the Apache configurations and use a browser to access *E/P:81/test2/*. The system prompts you to enter the authentication information, as shown in the following figure.



The image shows a 'Sign in' dialog box. At the top left is the text 'Sign in'. Below it is the URL 'http://120.46.46.115'. A warning message 'Your connection to this site is not private' is displayed. There are two input fields: 'Username' and 'Password', both with placeholder text. At the bottom are two buttons: a blue 'Sign in' button and a white 'Cancel' button.

Enter the user name **test** and password **Huawei@123**. The static resources can be accessed successfully, as shown in the following figure.



The image shows a web browser displaying the directory index for '/test2'. The title is 'Index of /test2'. The table has three columns: 'Name', 'Last modified', and 'Size Description'. It lists two files: 'Parent Directory' and 'test1', 'test2'. Both files have a size of 12 and were last modified on the same day.

Name	Last modified	Size Description
<a href="#">Parent Directory</a>		-
<a href="#">test1</a>		12
<a href="#">test2</a>		12

## 1.6.4 Quiz

Can the configuration of controlling static resource access permissions through accounts be written to **source.conf** instead of **.htaccess**?

Yes, it can be controlled by **AllowOverride**.

## 1.7 Configuring Apache Virtual Hosts

Requirements: Access the page of **test1** through **www.test1.com**, the page of **test2** through **www.test2.com**, and the page of **test3** through **www.test3.com**. Resolve the three domain names to **192.168.1.11**. Store **test1**, **test2**, and **test3** in **/home/source/test1**, **/home/source/test2**, and **/home/source/test3**, and set their page contents to "**hello,test1**", "**hello,test2**", and "**hello,test3**", respectively.

Step 1    Prepare resources.

Run the following commands to create three directories in **/home/source**:

```
cd /home/source  
mkdir test{1..3}
```

Create the corresponding files in the directories and enter the corresponding contents.

```
echo "hello,test1" > test1/test1  
echo "hello,test2" > test2/test2  
echo "hello,test3" > test3/test3
```

## Step 2 Create virtual hosts.

Create a **vhost.conf** configuration file in the **/etc/httpd/conf.d** directory and enter the following content:

```
<VirtualHost *:81>  
ServerName www.test1.com  
DocumentRoot "/home/source/test1"  
DirectoryIndex test1  
</VirtualHost>  
<VirtualHost *:81>  
ServerName www.test2.com  
DocumentRoot "/home/source/test2"  
DirectoryIndex test2  
</VirtualHost>  
<VirtualHost *:81>  
ServerName www.test3.com  
DocumentRoot "/home/source/test3"  
DirectoryIndex test3  
</VirtualHost>
```

## Step 3 Verify the configurations.

There is no DNS server in the environment. Therefore, you need to manually configure domain name resolution by adding the following content to the **/etc/hosts** file:

```
192.168.1.11 www.test1.com  
192.168.1.11 www.test2.com  
192.168.1.11 www.test3.com
```

Reload the configurations and run the following commands to perform a test:

```
curl www.test1.com:81  
curl www.test2.com:81  
curl www.test3.com:81
```

If the following information is displayed.

```
[root@apache-http conf.d]# curl www.test1.com:81
hello,test1
[root@apache-http conf.d]# curl www.test2.com:81
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="http://www.test2.com/test2/">here</a>.</p>
</body></html>
[root@apache-http conf.d]# curl www.test3.com:81
hello,test3
```

- Question: Why does **www.test2.com:81** not return the expected content?

Answer: The alias of **test2** is configured in the **source.conf** file. As a result, the system does not return the expected content. Comment out the alias in the **source.conf** file and try again. The expected content is returned, as shown in the following figure.

```
[root@apache-http conf.d]# curl www.test2.com:81
hello,test2
[root@apache-http conf.d]#
```

- Question: Is there any log in **error\_log** when an error is reported during the access to **www.test2.com:81**?

Answer: No, there is no such log. The **/etc/httpd/logs/error\_log** file does not store error logs of virtual hosts, but the **access\_log** file stores access logs of virtual hosts. To view related logs, add the following configuration to the virtual host:

```
errorLog "logs/test1_error_log"
```

## 1.8 Configuring the HTTPS Service

Set the access mode of **www.test1.com** to HTTPS based on section "1.7 Configuring Apache Virtual Hosts."

Step 1     Install the dependent module.

Run the following command to install mod\_ssl:

```
yum install -y mod_ssl
```

```
[root@apache-http conf.d]# yum install -y mod_ssl
local
Dependencies resolved.
=====
 Package                               Architec
=====
Installing:
 mod_ssl                               x86_64
Installing dependencies:
 sscg                                 x86_64

Transaction Summary
=====
Install 2 Packages

Total size: 149 k
Installed size: 367 k
Downloading Packages:
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing :
  Running scriptlet: sscg-3.0.1-1.oe2203.x86_64
  Installing   :
  Running scriptlet: sscg-3.0.1-1.oe2203.x86_64
  Running scriptlet: sscg-3.0.1-1.oe2203.x86_64
  Installing   :
  Running scriptlet: mod_ssl-1:2.4.51-5.oe2203.x86_64
  Running scriptlet: mod_ssl-1:2.4.51-5.oe2203.x86_64
  Verifying    :
  Verifying    : mod_ssl-1:2.4.51-5.oe2203.x86_64
  Verifying    : sscg-3.0.1-1.oe2203.x86_64

Installed:
 mod_ssl-1:2.4.51-5.oe2203.x86_64

Complete!
```

## Step 2 Create a self-signed certificate.

Run the following commands to create a private key:

```
mkdir ca
cd ca
openssl genrsa -out ca.key 2048
```

```
[root@apache-http ca]# openssl genrsa -out ca.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
...+++++
e is 65537 (0x010001)
```

Use the private key to generate a CA certificate.

```
openssl req -newkey rsa:4096 -nodes -sha256 -keyout ca.key -x509 -days 3650 -out ca.crt
```

```
[root@apache-http ca]# openssl req -newkey rsa:4096 -nodes -sha256 -keyout ca.key -x509 -days 3650 -out ca.crt
Generating a RSA private key
.....+=====
.....+=====
writing new private key to 'ca.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:CN
State or Province Name (full name) [Some-State]:ZJ
Locality Name (eg, city) []:HZ
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:test1.com
Email Address []:
```

Generate a request file using the private key and CA certificate.

```
openssl req -newkey rsa:4096 -nodes -sha256 -keyout www.test1.com.key -out www.test1.com.csr
```

```
[root@apache-http ca]# openssl req -newkey rsa:4096 -nodes -sha256 -keyout www.test1.com.key -out www.test1.com.csr
Generating a RSA private key
.....+=====
.....+=====
writing new private key to 'www.test1.com.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:CN
State or Province Name (full name) [Some-State]:ZJ
Locality Name (eg, city) []:HZ
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:www.test1.com
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Issue a certificate to the server.

```
openssl x509 -req -days 36500 -in www.test1.com.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out www.test1.com.crt
```

```
[root@apache-http ca]# openssl x509 -req -days 36500 -in www.test1.com.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out www.test1.com.crt
Signature ok
subject=C = CN, ST = ZJ, L = HZ, O = Internet Widgits Pty Ltd, CN = www.test1.com
Getting CA Private Key
```

### Step 3    Specify a certificate file.

Run the following command to open the SSL configuration file:

```
vim /etc/httpd/conf.d/ssl.conf
```

Edit the file as follows.

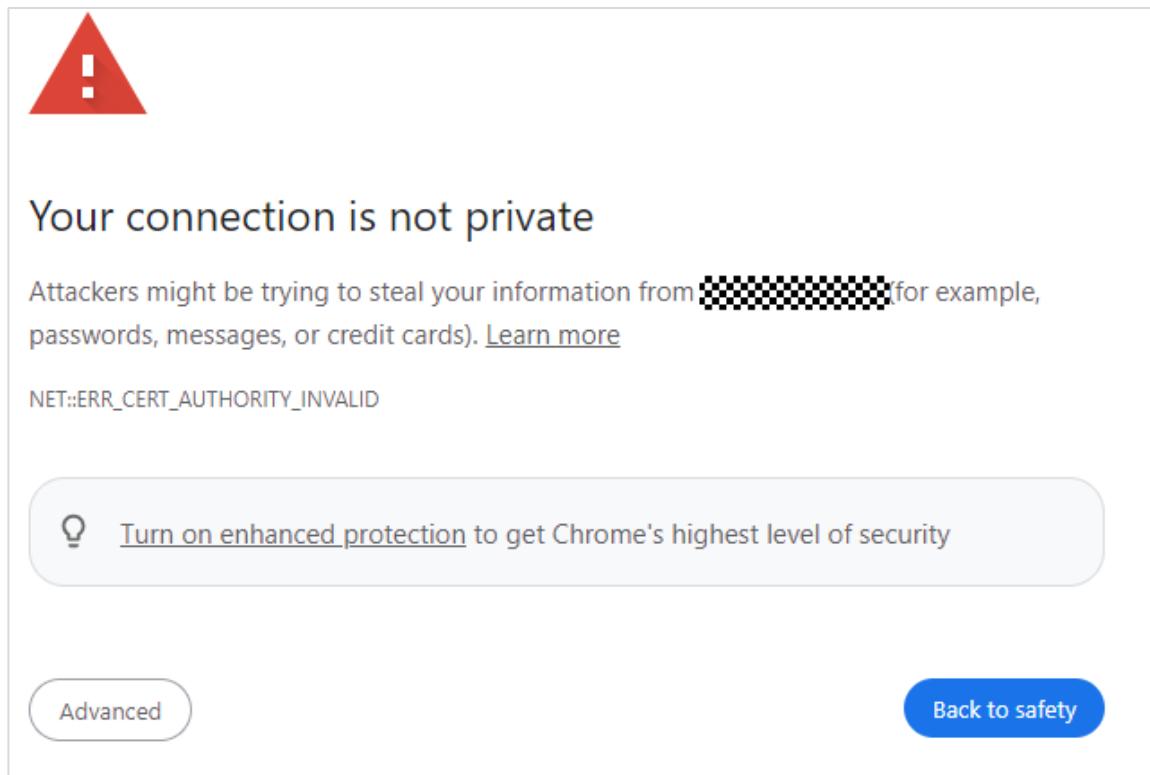
```
#SSLCertificateFile /etc/pki/tls/certs/localhost.crt
SSLCertificateFile /home/ca/www.test1.com.crt

# Server Private Key:
# If the key is not combined with the certificate, use this
# directive to point at the key file. Keep in mind that if
# you've both a RSA and a DSA private key you can configure
# both in parallel (to also allow the use of DSA ciphers, etc.)
# ECC keys, when in use, can also be configured in parallel
SSLCertificateKeyFile /home/ca/www.test1.com.key
#SSLCertificateKeyFile /etc/pki/tls/private/localhost.key
```

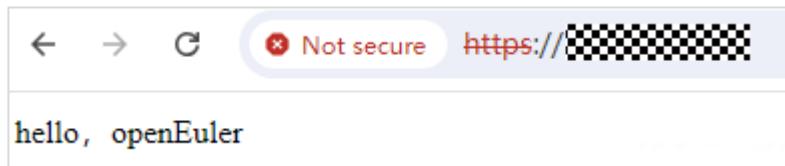
#### Step 4 Verify the configurations.

Add a static resolution entry to the client to resolve **www.test1.com** to the EIP of **Server01**.

Enter **https://www.test1.com** in the address box of the browser. The following page is displayed.



Click **Advanced** and select **Proceed to www.test1.com (unsafe)**. The following page is displayed.



- Question: Is the preceding page displayed after performing the preceding operations? Why?

Answer: The preceding page may not be displayed. If port 443 of **www.test1.com** is not configured on the virtual host, an error page is displayed.

# 2 NGINX Configurations

## 2.1 Installing and Testing NGINX

### 2.1.1 Preparing Resources

Purchase the **Server02** ECS as planned.

### 2.1.2 Procedure

Step 1    Install NGINX using the Yum repository.

In openEuler, run the following command to automatically install NGINX:

```
yum install -y nginx
```

```
Metadata cache created.
[root@localhost yum.repos.d]# yum install -y nginx
Last metadata expiration check: 0:00:23 ago on Thu Oct 5 14:20:44 2023 CST.
Dependencies resolved.
=====
 Package                                Architecture
=====
Installing:
nginx                                  x86_64
Installing dependencies:
gd                                     x86_64
gperftools-libs                         x86_64
libXpm                                 x86_64
libunwind                             x86_64
libwebp                               x86_64
libxslt                               x86_64
nginx-all-modules                      noarch
nginx-filesystem                       noarch
nginx-mod-http-image-filter            x86_64
nginx-mod-http-perl                   x86_64
nginx-mod-http-xslt-filter             x86_64
nginx-mod-mail                         x86_64
nginx-mod-stream                       x86_64
```

After the installation is complete, run the following command to check the NGINX version:

```
nginx -v
```

```
[root@localhost yum.repos.d]# nginx -v
nginx version: nginx/1.21.5
[root@localhost ~]
```

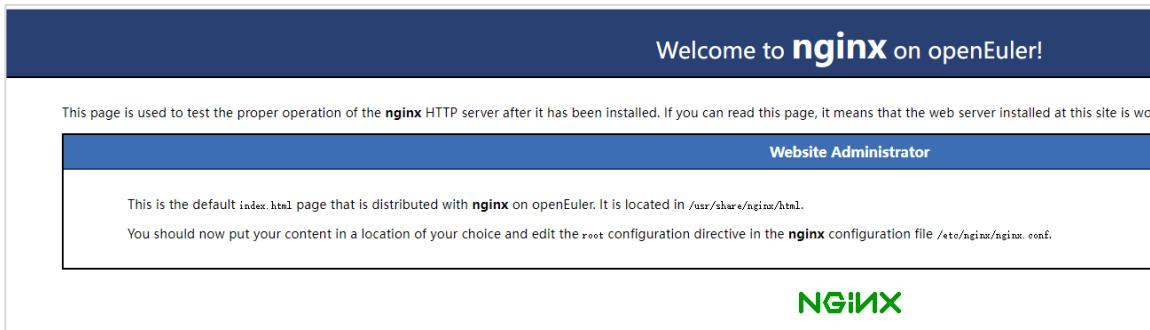
After confirming that the NGINX is correctly installed, run the following commands to start NGINX and check whether the NGINX processes are started:

```
systemctl start nginx  
ps -f | grep nginx
```

```
[root@localhost yum.repos.d]# systemctl start nginx  
[root@localhost yum.repos.d]# ps -ef | grep nginx  
root      9121     1  0 10:54 ?        00:00:00 nginx: master process /usr/sbin/nginx  
nginx    9122   9121  0 10:54 ?        00:00:00 nginx: worker process  
nginx    9123   9121  0 10:54 ?        00:00:00 nginx: worker process  
root     9125   5337  0 10:54 pts/0    00:00:00 grep --color=auto nginx
```

## Step 2 Verify the configurations.

Enter the IP address of the NGINX server in the address box of the browser. The following page is displayed.



## 2.2 Performing Basic NGINX Configurations

### 2.2.1 Configuring Static Resource Access

Description: Use NGINX to access different types of resources, such as HTML files, image files, and TXT files.

#### Step 1 Modify global configurations.

The default global configurations in the main configuration file are as follows:

```
user nginx;  
worker_processes auto;  
error_log /var/log/nginx/error.log;  
pid /run/nginx.pid;
```

Modify the configurations as follows:

1. Change the user of the worker process to **nobody**.
2. Change the number of worker processes to **4**.

The modified configurations are as follows:

```
user nobody;  
worker_processes 4;  
error_log /var/log/nginx/error.log;  
pid /run/nginx.pid;
```

Run the following commands to reload the NGINX configurations and check the processes:

```
nginx -s reload  
ps -ef | grep nginx
```

The following information is displayed.

```
[root@localhost nginx]# ps -ef | grep nginx  
root      9121      1  0 Feb09 ?        00:00:00 nginx: master process /usr/sbin/nginx  
nobody   13459    9121  0 14:28 ?        00:00:00 nginx: worker process  
nobody   13460    9121  0 14:28 ?        00:00:00 nginx: worker process  
nobody   13461    9121  0 14:28 ?        00:00:00 nginx: worker process  
nobody   13462    9121  0 14:28 ?        00:00:00 nginx: worker process  
root     13464    5337  0 14:29 pts/0    00:00:00 grep --color=auto nginx
```

The user of the worker processes changes to **nobody**, and the user of the master process changes to **root**. In addition, the number of worker processes changes to 4.

## Step 2 Access static web resources.

Requirement: Use the domain name **www.test.com** to access the specified home page of the NGINX service and the specified image file and TXT file.

Run the following command to create a directory for storing static resources:

```
mkdir -p /data/Nginx
```

See the following figure.

```
[root@localhost nginx]# mkdir -p /data/Nginx  
[root@localhost nginx]# ls /  
afs  bin  boot  data  dev  etc  home  lib  lib  
[root@localhost nginx]# ls /data  
Nginx
```

Create a home page file **index.html** in NGINX and enter "**hello,openEuler**".

```
touch /data/Nginx/index.html  
echo "hello,openEuler" > /data/Nginx/index.html
```

Create a **test.txt** file in the **Nginx** directory and enter "**hello,Nginx**".

```
touch /data/Nginx/test.txt  
echo "hello,Nginx" > /data/Nginx/test.txt
```

Copy **nginx-logo.png** in **/usr/share/Nginx/html** to the **Nginx** directory.

```
cp /usr/share/nginx/html/nginx-logo.png /data/Nginx/
```

After the configuration is complete, the files in the **Nginx** directory are as follows:

```
[root@localhost nginx]# ls /data/Nginx/  
index.html  nginx-logo.png  test.txt
```

In the **/etc/nginx/conf.d** directory, create a configuration file **static.conf** for the new static website and configure the following content:

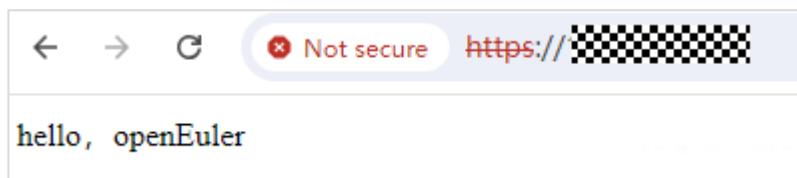
```
server {  
    listen 81;  
    server_name www.test.com;  
    root /data/Nginx;  
    index index.html;  
}
```

Run the **nginx -t** command to check whether the configuration is correct. If the configuration is incorrect, rectify the fault before performing subsequent operations. If the configuration is correct, the following information is displayed.

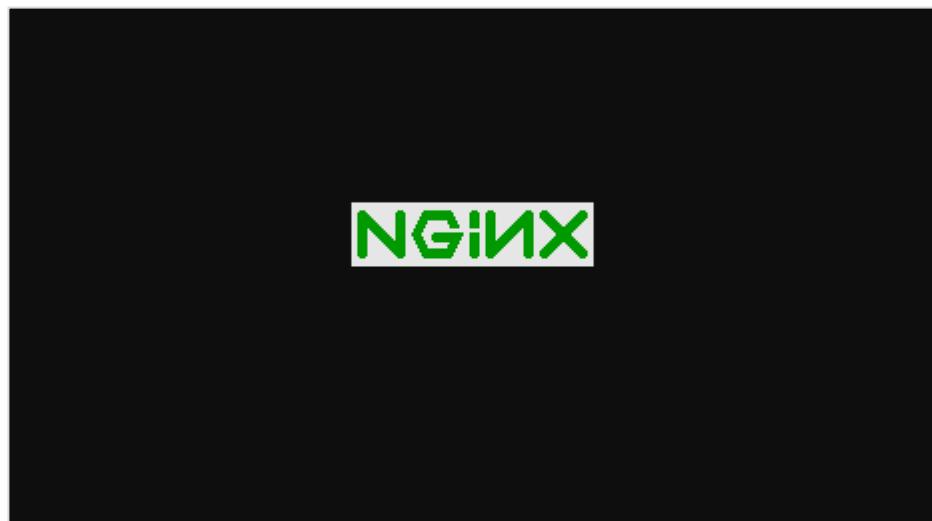
```
[root@localhost conf.d]# nginx -t  
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok  
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

After confirming that the information is correct, run the **nginx -s reload** command to reload the service.

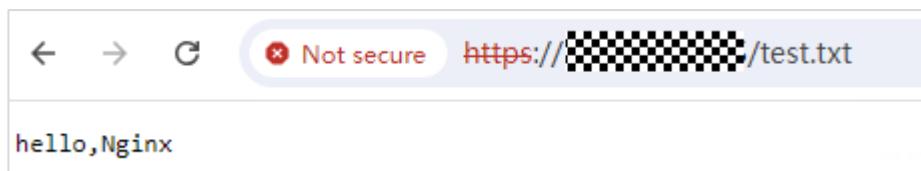
Configure static domain name resolution on the host where the browser is located to map **www.test.com** to the server. Then access **www.test.com:81** using the browser. The following page is displayed.



Access **www.test.com:81/nginx-logo.png**. The following page is displayed.



Access **www.test.com:81/test.txt**. The following page is displayed.



- Question 1: If a **test.nn** file is created in the **Nginx** directory and certain content is entered, will the system return the corresponding content when the file is accessed through **www.test.com:81/test.nn**? Why?

No, it will not. The **.nn** file is not in **mime.type**. Therefore, NGINX cannot parse the file. When you access the URL, the corresponding file is downloaded to the local host by default.

- Question 2: To implement the same function of the preceding website, is there any other writing format for the configuration file?

Yes, there is. You can refer to the following content:

```
server {  
    listen 81;  
    server_name www.test.com;  
    location / {  
        root /data/Nginx;  
        index index.html;  
    }  
}
```

## 2.2.2 Configuring Virtual Hosts

Description: On the NGINX server, configure virtual hosts using different port numbers.

### Step 1 Prepare resources.

Create the **nginx1**, **nginx2**, and **nginx3** directories in the **/data** directory.

```
mkdir nginx{1..3}
```

```
[root@localhost data]# mkdir nginx{1..3}  
[root@localhost data]# ls  
nginx1  nginx2  nginx3
```

Run the following commands to create **index.html** in the three directories and enter "hello,nginx1", "hello,nginx2", and "hello,nginx3" in the files:

```
echo "hello,nginx1" > nginx1/index.html  
echo "hello,nginx2" > nginx2/index.html  
echo "hello,nginx3" > nginx3/index.html
```

```
[root@localhost data]# echo "hello,nginx1" > nginx1/index.html  
[root@localhost data]# echo "hello,nginx2" > nginx2/index.html  
[root@localhost data]# echo "hello,nginx3" > nginx3/index.html  
[root@localhost data]# cat nginx1/index.html  
hello,nginx1  
[root@localhost data]# cat nginx2/index.html  
hello,nginx2  
[root@localhost data]# cat nginx3/index.html  
hello,nginx3
```

### Step 2 Configure virtual hosts.

To avoid interference between configurations, delete the existing configuration file in **/etc/nginx/conf.d** or change the file name extension to **.bk**.

Create a configuration file **vhost.conf** and enter the following content:

```
server {  
    listen 81;  
    server_name localhost;  
    location / {  
        root /data/nginx1;  
        index index.html;  
    }  
}  
server {  
    listen 82;  
    server_name localhost;  
    location / {  
        root /data/nginx2;  
        index index.html;  
    }  
}  
server {  
    listen 83;  
    server_name localhost;  
    location / {  
        root /data/nginx3;  
        index index.html;  
    }  
}
```

After the configuration is complete, run the **nginx -t** command to check whether syntax errors exist. If no syntax error exists, run the **nginx -s reload** command to reload the configurations.

### Step 3 Verify the configurations.

Run the **curl** command to configure virtual hosts with different ports, as shown in the following figure.

```
[root@localhost conf.d]# curl localhost:81  
hello,nginx1  
[root@localhost conf.d]# curl localhost:82  
hello,nginx2  
[root@localhost conf.d]# curl localhost:83  
hello,nginx3
```

## 2.2.3 Configuring the Location Directive

Description: This lab is based on the virtual host configuration. Try to use different representation methods to configure routes.

### Step 1 Prepare resources.

Rename **/data/nginx1** to **/data/Nginx1**.

### Step 2 Modify the configuration file.

Modify the **vhost.conf** configuration file as follows:

```
server {  
    listen 81;
```

```
server_name localhost;
location / {
    root /data/nginx1;
    index index.html;
}
}
server {
listen 82;
server_name localhost;
location /nginx2 {
    root /data;
    index index.html;
}
}
server {
listen 83;
server_name localhost;
location / {
    root /data/nginx3;
    index index.html;
}
}
```

Reload the NGINX configurations.

### Step 3 Verify the configurations.

Run the **curl localhost:81**, **curl localhost:82**, and **curl localhost:83** commands. The following information is displayed.

```
[root@localhost conf.d]# curl localhost:81
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx/1.21.5</center>
</body>
</html>
[root@localhost conf.d]# curl localhost:82
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Test Page for the Nginx HTTP Server on openEuler</title>
    <style type="text/css">
        /*<![CDATA[*/
        body {
            background-color: #fff;
            color: #000;
            font-size: 0.9em;
            font-family: sans-serif, helvetica;
            margin: 0;
            padding: 0;
        }
        :link {
[root@localhost conf.d]# curl localhost:83
hello,nginx3</pre>
```

After you run the **curl localhost:82/nginx2** command, the following information is displayed.

```
[root@localhost conf.d]# curl localhost:82/nginx2
<html>
<head><title>301 Moved Permanently</title></head>
<body>
<center><h1>301 Moved Permanently</h1></center>
<hr><center>nginx/1.21.5</center>
</body>
</html>
```

- Question: What conclusions can be drawn based on the preceding information?

Answer:

1. NGINX is case sensitive. After **nginx1** is changed to **Nginx1**, the corresponding resource cannot be accessed.
2. After the **location** directive specifies a path, the specified path will be matched in the URL when the resource is accessed.
3. If the resource to be accessed is a directory, a slash (/) needs to be added to the end of the URL.

#### Step 4 Modify the configuration file.

Modify the **vhost.conf** file as follows:

```
server {
    listen 81;
    server_name localhost;
    location ~* \.html$ {
        root /data/Nginx1;
        #   index index.html;
    }
}
server {
    listen 82;
    server_name localhost;
    location = /nginx2/index.html {
        root /data;
        #   index index.html;
    }
}
server {
    listen 83;
    server_name localhost;
    location = / {
        root /data/nginx3;
        index index.html;
    }
}
```

Reload the NGINX configurations.

#### Step 5 Verify the configurations.

- Question: Based on the preceding configuration file, which URLs can be used to access the corresponding resources?

Answer: Virtual host 1 uses a case-insensitive regular expression to match all resources whose name extension is **html**. The corresponding URL is **localhost:81**. See the following figure.

```
[root@localhost conf.d]# curl localhost:81  
hello,nginx1
```

Virtual host 2 uses exact match to strictly match the required resource. The corresponding URL is **localhost:82/nginx2/index.html**. See the following figure.

```
[root@localhost conf.d]# curl localhost:82/nginx2/index.html  
hello,nginx2
```

Virtual host 3 also uses exact match but the URL must end with a slash (/). However, the actual path of the server is **/data/nginx3/index.html**. For this reason, the corresponding resource cannot be accessed, and the system returns the default home page. See the following figure.

```
[root@localhost conf.d]# curl localhost:83  
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml11.dtd">  
  
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">  
  <head>  
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />  
    <title>Test Page for the Nginx HTTP Server on openEuler</title>  
    <style type="text/css">  
      /*<![CDATA[*/  
      body {  
        background-color: #fff;  
        color: #000;  
        font-size: 0.9em;  
        font-family: sans-serif,helvetica;  
        margin: 0;  
        padding: 0;  
      }  
      :link {  
        color: #c00;  
      }  
      :visited {  
        color: #c00.  
      }
```

## Step 6 Modify the configuration file.

Modify the **vhost.conf** file as follows:

```
server {  
  listen 81;  
  server_name localhost;  
  location /Nginx1 {  
    root /data;  
    index index.html;  
  }  
}  
server {  
  listen 82;  
  server_name localhost;  
  location /nginx2 {  
    alias /data/nginx2/index.html;  
    #    index index.html;  
  }  
}  
server {  
  listen 83;  
  server_name localhost;  
  location = / {
```

```
root /data/nginx3;
index index.html;
}
```

Reload the NGINX configurations.

#### Step 7 Verify the configurations.

Run the **curl localhost:81/Nginx1/** and **curl localhost:82/nginx2** commands. The following information is displayed.

```
[root@localhost conf.d]# curl localhost:81/Nginx1/
hello,nginx1
[root@localhost conf.d]# curl localhost:82/nginx2
hello,nginx2
```

- Question: What conclusions can be drawn from the preceding test?

Answer: When **root** is used to specify a resource path in **location**, a relative path is used. When an alias is used, an absolute path is used.

## 2.3 Configuring Reverse Proxies and Load Balancing with NGINX

### 2.3.1 Configuring Reverse Proxies with NGINX

Description: This lab describes how to configure and use the reverse proxy function of NGINX based on section "Configuring the Location Directive". However, it is greatly different from the actual application scenario. You are advised not to use this architecture in actual practice.

#### Step 1 Set virtual host 1 as the proxy of virtual host 3.

Modify the **vhost.conf** file as follows:

```
server {
    listen 81;
    server_name localhost;
    location / {
        proxy_pass http://127.0.0.1:83;
    }
}
server {
    listen 82;
    server_name localhost;
    location /nginx2 {
        alias /data/nginx2/index.html;
    }
}
```

```
server {
    listen 83;
    server_name localhost;
    location / {
        root /data/nginx3;
        index index.html;
    }
}
```

Reload the NGINX configurations.

#### Step 2 Verify the configurations.

Run the **curl localhost:81** command to access the corresponding resource and check the returned information, as shown in the following figure.

```
[root@localhost conf.d]# curl localhost:81
hello,nginx3
```

#### Step 3 Set virtual host 1 as the proxy of virtual host 2.

Modify the **vhost.conf** file as follows:

```
server {
    listen 81;
    server_name localhost;
    location /nginx1 {
        proxy_pass http://127.0.0.1:82/nginx2;
        #   root /data;
        #   index index.html;
    }
}
server {
    listen 82;
    server_name localhost;
    location /nginx2 {
        alias /data/nginx2/index.html;
        #   index index.html;
    }
}
server {
    listen 83;
    server_name localhost;
    location / {
        root /data/nginx3;
        index index.html;
    }
}
```

After the configuration is complete, reload the NGINX configurations.

#### Step 4 Verify the configurations.

Run the **curl localhost:81/nginx1** command to access the corresponding resource and check the returned information, as shown in the following figure.

```
[root@localhost conf.d]# curl localhost:81/nginx1
hello,nginx1
[root@localhost conf.d]# curl localhost:82/nginx2
hello,nginx2
[root@localhost conf.d]# curl localhost:83/nginx3
hello,nginx3
```

## 2.3.2 Configuring Load Balancing with NGINX

Description: This lab describes how to configure and use NGINX virtual hosts for load balancing. However, it is greatly different from the actual application scenario. You are advised not to use this architecture in actual practice.

### Step 1 Prepare resources.

Restore the **vhost.conf** file to the content in step 2 in section "Configuring Virtual Hosts." The details are as follows:

```
server {
    listen 81;
    server_name localhost;
    location / {
        root /data/nginx1;
        index index.html;
    }
}
server {
    listen 82;
    server_name localhost;
    location / {
        root /data/nginx2;
        index index.html;
    }
}
server {
    listen 83;
    server_name localhost;
    location / {
        root /data/nginx3;
        index index.html;
    }
}
```

Restore **Nginx1** to **nginx1**. After the restoration, the **/data** directory is as follows.

```
[root@localhost conf.d]# ls /data/
nginx1  nginx2  nginx3
```

Reload the NGINX configurations and check whether the services corresponding to the virtual hosts can be accessed, as shown in the following figure.

```
[root@localhost data]# curl 192.168.1.12:81
hello,nginx1
[root@localhost data]# curl 192.168.1.12:82
hello,nginx2
[root@localhost data]# curl 192.168.1.12:83
hello,nginx3
```

### Step 2 Configure load balancing.

Create a load balancing configuration file **lb.conf** in the **conf.d** directory and enter the following content:

```
upstream www.test.com {  
    server 192.168.1.12:81;  
    server 192.168.1.12:82;  
    server 192.168.1.12:83;  
}  
server {  
    location / {  
        proxy_pass http://www.test.com;  
    }  
}
```

After the configuration is complete, reload the NGINX configurations and manually add the static domain name resolution of **www.test.com** to **/etc/hosts**, as shown in the following figure.

```
127.0.0.1 localhost localhost.localdomain localhost  
::1 localhost localhost.localdomain localhost  
192.168.1.12 www.test.com
```

### Step 3 Verify the configurations.

Run the **curl www.test.com** command on the server for multiple times and check the returned information, as shown in the following figure.

```
[root@localhost conf.d]# curl www.test.com  
hello,nginx1  
[root@localhost conf.d]# curl www.test.com  
hello,nginx2  
[root@localhost conf.d]# curl www.test.com  
hello,nginx3  
[root@localhost conf.d]# curl www.test.com  
hello,nginx1  
[root@localhost conf.d]# curl www.test.com  
hello,nginx2  
[root@localhost conf.d]# curl www.test.com  
hello,nginx3
```

It can be observed that each request is served by the three virtual hosts in turn.

### Step 4 Configure weights.

Modify the **lb.conf** file to add weights for load balancing as follows:

```
upstream www.test.com {  
    server 192.168.1.12:81 weight=2;  
    server 192.168.1.12:82 weight=1;  
    server 192.168.1.12:83;  
}  
server {  
    location / {  
        proxy_pass http://www.test.com;  
    }  
}
```

}

Reload the NGINX configurations.

#### Step 5 Verify the configurations.

Run the **curl www.test.com** command on the server for multiple times and check the returned information, as shown in the following figure.

```
[root@localhost conf.d]# curl www.test.com
hello,nginx1
[root@localhost conf.d]# curl www.test.com
hello,nginx1
[root@localhost conf.d]# curl www.test.com
hello,nginx2
[root@localhost conf.d]# curl www.test.com
hello,nginx3
[root@localhost conf.d]# curl www.test.com
hello,nginx1
[root@localhost conf.d]# curl www.test.com
hello,nginx1
[root@localhost conf.d]# curl www.test.com
hello,nginx2
[root@localhost conf.d]# curl www.test.com
hello,nginx3
```

The three virtual hosts provide services based on the configured weights.

#### Step 6 Configure the backup NGINX server.

Modify the **lb.conf** configuration file as follows:

```
upstream www.test.com {
    server 192.168.1.12:81 backup;
    server 192.168.1.12:82;
    server 192.168.1.12:83;
}
server {
    location / {
        proxy_pass http://www.test.com;
    }
}
```

After the configuration is complete, reload the NGINX configurations.

#### Step 7 Verify the configurations.

Run the **curl www.test.com** command on the server for multiple times and check the returned information, as shown in the following figure.

```
[root@localhost conf.d]# curl www.test.com
hello,nginx2
[root@localhost conf.d]# curl www.test.com
hello,nginx3
[root@localhost conf.d]# curl www.test.com
hello,nginx2
[root@localhost conf.d]# curl www.test.com
hello,nginx3
```

You can see that NGINX does not forward requests to virtual host 1.

#### Step 8    Enable the backup host.

Modify the **lb.conf** configuration file as follows:

```
upstream www.test.com {  
    server 192.168.1.12:81 backup;  
    server 192.168.1.12:82 down;  
    server 192.168.1.12:83 down;  
}  
server {  
    location / {  
        proxy_pass http://www.test.com;  
    }  
}
```

Reload the NGINX configurations.

#### Step 9    Verify the configurations.

Run the **curl www.test.com** command on the server for multiple times and check the returned information, as shown in the following figure.

```
[root@localhost conf.d]# curl www.test.com  
hello,nginx1  
[root@localhost conf.d]# curl www.test.com  
hello,nginx1  
[root@localhost conf.d]# curl www.test.com  
hello,nginx1
```

- Question: If you change the **lb.conf** file to the following content:

```
upstream www.test.com {  
    server 192.168.1.12:81 backup;  
    server 192.168.1.12:82;  
    server 192.168.1.12:83 down;  
}  
server {  
    location / {  
        proxy_pass http://www.test.com;  
    }  
}
```

What is returned by the system when you run the **curl www.test.com** command to request a resource? Why?

Answer: The system returns "**hello,nginx2**". Because virtual host 3 is down but virtual host 2 is still available, NGINX does not forward the request to backup virtual host 1 and the system thus returns the resource provided by virtual host 2. See the following figure.

```
[root@localhost conf.d]# curl www.test.com
hello,nginx2
[root@localhost conf.d]# curl www.test.com
hello,nginx2
[root@localhost conf.d]# curl www.test.com
hello,nginx2
[1]+ 11 SIGHUP
```

# 3 DNS Configurations

## 3.1 Introduction

### 3.1.1 About This Lab

This lab is based on section "Configuring Load Balancing with NGINX." The DNS server is used to replace the original static DNS to provide services for the client.

## 3.2 Installing the DNS Software

### 3.2.1 Preparing Resources

Purchase the **Server03** ECS as planned.

### 3.2.2 Installing the DNS Software

Run the following command on the newly created VM that functions as the DNS server to install DNS-related software:

```
yum install -y bind
```

Wait until the installation is complete and run the following command to start DNS-related services:

```
systemctl start named
```

If no error message is displayed, check whether the port used by DNS is listened on, as shown in the following figure.

```
[root@localhost yum.repos.d]# ss -lnp | grep 53
u_str LISTEN 0      4096          /run/systemd/private 18853
          * 0          users:(("systemd",pid=1,fd=15))
udp  UNCONN 0      0            127.0.0.1:53
          0.0.0.0:*  users:(("named",pid=9119,fd=24))
udp  UNCONN 0      0            127.0.0.1:53
          0.0.0.0:*  users:(("named",pid=9119,fd=25))
udp  UNCONN 0      0            [::]:53
          [::]:*   users:(("named",pid=9119,fd=31))
udp  UNCONN 0      0            [::]:53
          [::]:*   users:(("named",pid=9119,fd=30))
tcp  LISTEN 0      10           127.0.0.1:53
          0.0.0.0:*  users:(("named",pid=9119,fd=26))
tcp  LISTEN 0      10           127.0.0.1:53
          0.0.0.0:*  users:(("named",pid=9119,fd=27))
tcp  LISTEN 0      4096         127.0.0.1:953
          0.0.0.0:*  users:(("named",pid=9119,fd=23))
tcp  LISTEN 0      10           [::]:53
          [::]:*   users:(("named",pid=9119,fd=33))
tcp  LISTEN 0      10           [::]:53
          [::]:*   users:(("named",pid=9119,fd=32))
tcp  LISTEN 0      4096         [::]:953
          [::]:*   users:(("named",pid=9119,fd=34))
```

If port 53 is listened on, DNS is installed and started successfully.

### 3.3 Setting Up the Master DNS Server

Resolve **www.test.com** to **192.168.1.12** on the newly created DNS server.

#### Step 1 Create a forward resolution resource record.

Bind provides the administrator with a template file for configuring resource records. The file is stored in the **/var/named** directory and named **named.localhost**. Run the **cp** command to copy the file as **test.com.zone**. The details are as follows:

```
cd /var/named  
cp named.localhost test.com.zone -p
```

After the copy is complete, modify the configurations in **test.com.zone** as follows:

```
$TTL 1D  
@ IN SOA master.test.com. admin.test.com. (  
          0 ; serial  
          1D ; refresh  
          1H ; retry  
          1W ; expire  
          3H ) ; minimum  
NS master  
master A 192.168.1.13  
www CNAME main  
main A 192.168.1.12
```

#### Step 2 Modify the configurations.

Modify the listening port and permission configurations in the **/etc/named.conf** configuration file of the DNS service, as shown in the following figure.

```
options {  
    listen-on port 53 { localhost; };  
    listen-on-v6 port 53 { ::1; };  
    directory      "/var/named";  
    dump-file     "/var/named/data/cache_dump.db";  
    statistics-file "/var/named/data/named_stats.txt";  
    memstatistics-file "/var/named/data/named_mem_stats.txt";  
    secroots-file  "/var/named/data/named.secroots";  
    recursing-file "/var/named/data/named.recur sing";  
    allow-query    { any; };  
};  
/*
```

Add the **test.com zone** configuration to **/etc/named.rfc1912.zones**, as shown in the following figure.

```
zone "test.com" IN {
    type master;
    file "test.com.zone";
};

zone "localhost.localdomain" IN {
    type master;
    file "named.localhost";
    allow-update { none; };
};
```

After the preceding configurations are complete, run the **named-checkconf** command to check whether the DNS configuration file is correct, and run the **named-checkzone test.com /var/named/test.com.zone** command to check whether the **test.com zone** configuration is correct, as shown in the following figure.

```
[root@localhost named]# named-checkconf
[root@localhost named]# named-checkzone test.com /var/named/test.com.zone
zone test.com/IN: loaded serial 0
OK
```

If all configurations pass the check, run the **rndc reload** command to reload the DNS service configurations, as shown in the following figure.

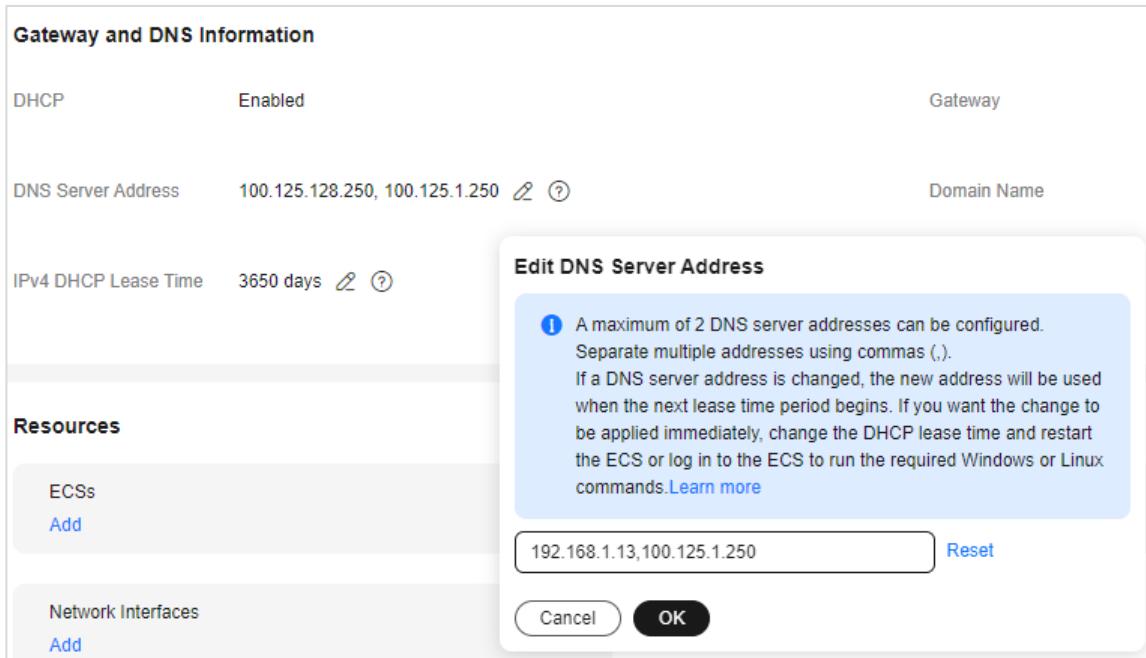
```
[root@localhost named]# rndc reload
server reload successful
```

### Step 3 Verify the configurations.

On the VM of the NGINX server, run the following commands (change **ens33** in the commands as required) to point the DNS server address to the newly configured server:

```
nmcli con mod ens33 ipv4.dns 192.168.1.13
nmcli con down ens33 && nmcli con up ens33
```

To ensure the lab effect, manually set the DNS server address of the VPC where the ECS resides to **192.168.1.13**, as shown in the following figure.



DHCP Enabled

DNS Server Address 100.125.128.250, 100.125.1.250

IPv4 DHCP Lease Time 3650 days

Resources

ECSs

Network Interfaces

Gateway

Domain Name

Edit DNS Server Address

A maximum of 2 DNS server addresses can be configured. Separate multiple addresses using commas (,). If a DNS server address is changed, the new address will be used when the next lease time period begins. If you want the change to be applied immediately, change the DHCP lease time and restart the ECS or log in to the ECS to run the required Windows or Linux commands.[Learn more](#)

192.168.1.13,100.125.1.250

Cancel OK

Reset

After the configuration is complete, run the **nslookup** command to check whether the DNS service can correctly resolve **www.test.com** to **192.168.1.12**, as shown in the following figure.

```
[root@localhost ~]# nslookup www.test.com
Server:      192.168.1.13
Address:     192.168.1.13#53

www.test.com    canonical name = main.test.com.
Name:   main.test.com
Address: 192.168.1.12
```

Run the **curl** command to access **www.test.com**, as shown in the following figure.

```
[root@localhost ~]# curl www.test.com
hello,nginx2
```

## 3.4 Configuring Reverse DNS Resolution

Resolve **www.test.com** to **192.168.1.12** on the newly created DNS server.

Step 1 Create a reverse resolution resource record.

Bind provides the administrator with a template file for configuring resource records. The file is stored in the **/var/named** directory and named **named.loopback**. Run the **cp** command to copy the file as **192.168.1.zone**. The details are as follows:

```
cd /var/named
cp named.loopback 192.168.1.zone -p
```

After the copy is complete, modify the configurations in **192.168.1.zone** as follows:

```
$TTL 1D
@ IN SOA master.test.com. admin.test.com. (
                                0      ; serial
                                1D     ; refresh
                                1H     ; retry
                                1W     ; expire
                                3H )   ; minimum
NS      master
master A 192.168.1.13
12    PTR  www.test.com
```

## Step 2 Modify the configurations.

Add the following configurations to the **/etc/named.rfc1912.zones** file.

```
zone "1.0.0.127.in-addr.arpa" IN {
        type master;
        file "named.loopback";
        allow-update { none; };
};

zone "0.in-addr.arpa" IN {
        type master;
        file "named.empty";
        allow-update { none; };
};

zone "1.168.192.in-addr.arpa" IN {
        type master;
        file "192.168.1.zone";
        allow-update { none; };
};
```

Check whether the configurations are correct, as shown in the following figure.

```
[root@localhost named]# named-checkconf
[root@localhost named]# named-checkzone 1.168.192 /var/named/192.168.1.zone
zone 1.168.192/IN: loaded serial 0
OK
```

If no error is found, run the **rndc reload** command to reload the DNS configuration file.

```
[root@localhost named]# rndc reload
server reload successful
```

## Step 3 Verify the configurations.

Log in to the NGINX server and run the **nslookup** command to check whether the configurations take effect.

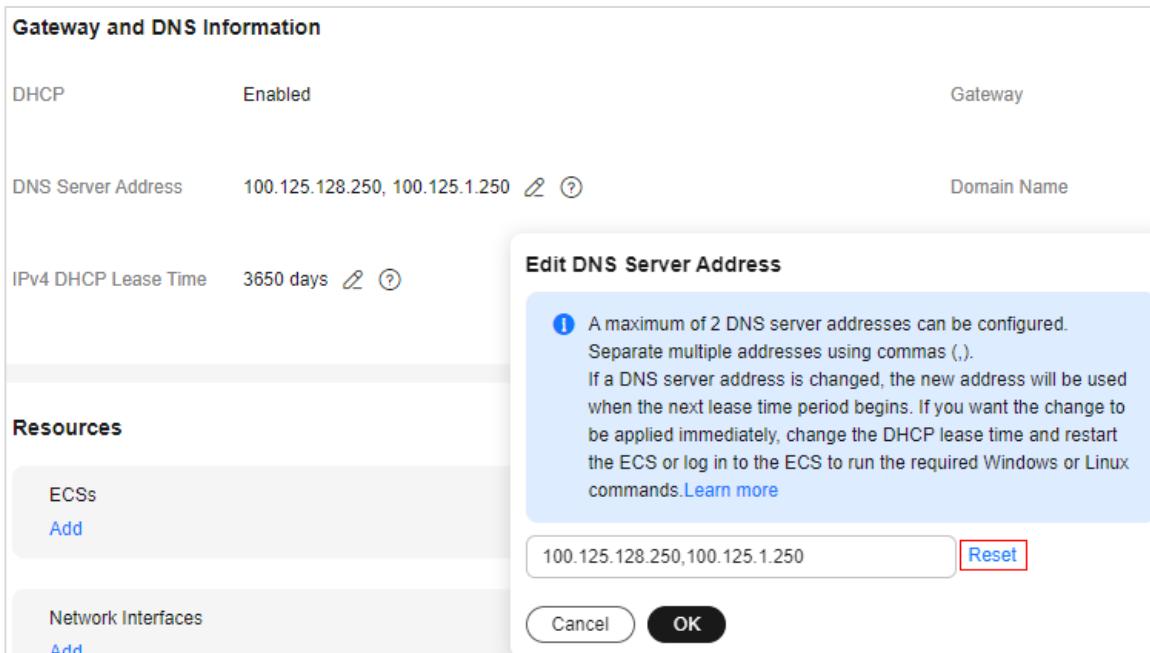
```
server reload successful
[root@localhost named]# nslookup
> 192.168.1.12
12.1.168.192.in-addr.arpa      name = www.test.com.1.168.192.in-addr.arpa.
```

# 4 MySQL Configurations

## 4.1 Installing and Initializing MySQL

### 4.1.1 Preparing Resources

Purchase the **Server04** ECS as planned, and reset the VPC DNS configurations modified in the preceding lab. Otherwise, the ECS cannot connect to the network.



**Gateway and DNS Information**

DHCP	Enabled	Gateway
DNS Server Address	100.125.128.250, 100.125.1.250	Domain Name
IPv4 DHCP Lease Time	3650 days	

**Resources**

ECGs	Add
Network Interfaces	Add

**Edit DNS Server Address**

A maximum of 2 DNS server addresses can be configured. Separate multiple addresses using commas (,). If a DNS server address is changed, the new address will be used when the next lease time period begins. If you want the change to be applied immediately, change the DHCP lease time and restart the ECS or log in to the ECS to run the required Windows or Linux commands.[Learn more](#)

100.125.128.250,100.125.1.250 Reset

Cancel OK

### 4.1.2 Installing MySQL

Step 1 Check the MySQL version supported by the Yum repository of openEuler.

Run the following command on openEuler to check the MySQL version supported by the Yum repository of openEuler:

```
 yum provides mysql-server*
```



```
[root@localhost yum.repos.d]# yum provides mysql-server*
Last metadata expiration check: 0:33:10 ago on Thu Jul 10 10:45:20 2014 AM CST.
mysql-server-8.0.28-1.oe2203.x86_64 : The MySQL server and related files
Repo      : @System
Matched from:
Provide   : mysql-server = 8.0.28-1.oe2203
Provide   : mysql-server(x86-64) = 8.0.28-1.oe2203

mysql-server-8.0.28-1.oe2203.x86_64 : The MySQL server and related files
Repo      : everything
Matched from:
Provide   : mysql-server(x86-64) = 8.0.28-1.oe2203
Provide   : mysql-server = 8.0.28-1.oe2203

mysql5-server-5.7.34-2.oe2203.x86_64 : The MySQL server and related files
Repo      : everything
Matched from:
Provide   : mysql-server = 5.7.34-2.oe2203

mysql5-server-5.7.37-1.oe2203.x86_64 : The MySQL server and related files
Repo      : update
Matched from:
Provide   : mysql-server = 5.7.37-1.oe2203

mysql5-server-5.7.38-1.oe2203.x86_64 : The MySQL server and related files
Repo      : update
Matched from:
Provide   : mysql-server = 5.7.38-1.oe2203
```

According to the query result, you can install MySQL 8.0.x or 5.7.x.

## Step 2     Install MySQL 8.0.28.

Run the **yum** command to install MySQL-server, as shown in the following figure.

```
[root@localhost yum.repos.d]# yum install -y mysql-server
OS                                         703 kB/s | 3.4 MB   00:04
everything                                    3.7 MB/s | 16 MB    00:04
EPOL                                         2.9 MB/s | 2.6 MB   00:00
debuginfo                                     3.0 MB/s | 3.9 MB   00:01
source                                         2.2 MB/s | 1.7 MB   00:00
update                                         2.9 MB/s | 34 MB    00:11
Last metadata expiration check: 0:00:01 ago on Thu 23 Feb 2023 09:43:37 AM CST.
Dependencies resolved.

=====
Package           Architecture Version       Repository  Size
=====
Installing:
mysql-server      x86_64      8.0.28-1.oe2203  everything  23 M
Installing dependencies:
checkpolicy        x86_64      3.3-1.oe2203   OS          290 k
mariadb-config     x86_64      4:10.5.16-1.oe2203 update      8.0 k
mecab              x86_64      0.996-2.oe2203  everything  371 k
mysql              x86_64      8.0.28-1.oe2203  OS          10 M
mysql-common       x86_64      8.0.28-1.oe2203  OS          30 k
mysqlerrmsg        x86_64      8.0.28-1.oe2203  everything  545 k
mysql-selinux       noarch      1.0.0-2.oe2203  everything  35 k
policycoreutils-python-utils      noarch      3.3-1.oe2203   OS          25 k
protobuf-lite       x86_64      3.14.0-6.oe2203 update      217 k
python3-IPy         noarch      1.0-1.oe2203   OS          37 k
```

After the installation is complete, run the `systemctl start mysqld` command to start the service and check whether the corresponding port is listened on, as shown in the following figure.

### 4.1.3 Logging In to and Initializing MySQL

#### Step 1 Log in to MySQL.

After the installation is complete, run the **mysql** command to log in to the MySQL database, as shown in the following figure.

```
[root@localhost yum.repos.d]# mysql
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.28 Source distribution

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> █
```

- Question: What are the user name and password for logging in to the MySQL database?

Answer:

The current user is **root**. After 8.0.x is installed, the default password of user **root** is empty. Therefore, you can log in to the MySQL database without entering the password. However, the permission is low. You must initialize the password before performing the next step.

In MySQL 5.7, the password of user **root** is not empty and is automatically generated by the system. You can view the password in the **/var/log/mysql/mysqld.log** file.

#### Step 2 Initialize MySQL.

Run the following command to configure the password of user **root**:

```
alter user root@'localhost' identified by 'Huawei@123';
```

See the following figure.

```
mysql> alter user root@'localhost' identified by 'Huawei@123';
Query OK, 0 rows affected (0.00 sec)
```

After the modification is complete, exit the MySQL database and log in to the MySQL database as user **root** again.

```
mysql> alter user root@'localhost' identified by 'Huawei@123';
Query OK, 0 rows affected (0.00 sec)

mysql> exit
Bye
[root@localhost yum.repos.d]# mysql -uroot -p'Huawei@123'
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.28 Source distribution

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> █
```

Run the **show databases;** command to view the default databases, as shown in the following figure.

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
4 rows in set (0.01 sec)
```

Run the **select user,host from mysql.user;** command to view the information about the current system user, as shown in the following figure.

```
mysql> select user,host from mysql.user;
+-----+-----+
| user      | host     |
+-----+-----+
| mysql.infoschema | localhost |
| mysql.session    | localhost |
| mysql.sys        | localhost |
| root           | localhost |
+-----+-----+
4 rows in set (0.00 sec)
```

#### 4.1.4 Using mysqladmin

Step 1 Check the MySQL status and version.

Use the **status** statement to check the status of the MySQL database.

```
mysqladmin -uroot -p'Huawei@123' status
```

```
[root@localhost ~]# mysqladmin -uroot -p'Huawei@123' status
mysqladmin: [Warning] Using a password on the command line interface can be insecure.
Uptime: 22161 Threads: 2 Questions: 100 Slow queries: 0 Opens: 208 Flush tables: 3 Open tables: 124 Queries per second avg: 0.004
```

Use the **version** statement to view the MySQL version.

```
mysqladmin -uroot -p'Huawei@123' version
```

```
[root@localhost ~]# mysqladmin -uroot -p'Huawei@123' version
mysqladmin: [Warning] Using a password on the command line interface can be insecure.
mysqladmin Ver 8.0.28 for Linux on x86_64 (Source distribution)
Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Server version      8.0.28
Protocol version   10
Connection          Localhost via UNIX socket
UNIX socket        /var/lib/mysql/mysql.sock
Uptime:             6 hours 11 min 24 sec

Threads: 2  Questions: 102  Slow queries: 0  Opens: 208  Flush tables: 3  Open tables: 124  Queries per second avg: 0.004
```

Use the **processlist** statement to view the active threads of the MySQL database.

```
mysqladmin -uroot -p'Huawei@123' processlist
```

```
[root@localhost ~]# mysqladmin -uroot -p'Huawei@123' processlist
mysqladmin: [Warning] Using a password on the command line interface can be insecure.
+-----+-----+-----+-----+-----+-----+
| Id | User       | Host     | db    | Command | Time   | State           | Info
+-----+-----+-----+-----+-----+-----+
| 5  | event_scheduler | localhost |      | Daemon  | 22372 | Waiting on empty queue |
| 24 | root        | localhost |      | Query   | 0      | init            | show processlist
+-----+-----+-----+-----+-----+-----+
```

## Step 2 Flush MySQL tables and threads.

Run the following command to flush all tables:

```
mysqladmin -uroot -p'Huawei@123' flush-tables
```

Run the following command to flush the cache of all threads:

```
mysqladmin -uroot -p'Huawei@123' flush-threads
```

Run the following command to flush all logs:

```
mysqladmin -uroot -p'Huawei@123' flush-logs
```

```
[root@localhost ~]# mysqladmin -uroot -p'Huawei@123' flush-tables
mysqladmin: [Warning] Using a password on the command line interface can be insecure.
[root@localhost ~]# mysqladmin -uroot -p'Huawei@123' flush-threads
mysqladmin: [Warning] Using a password on the command line interface can be insecure.
[root@localhost ~]# mysqladmin -uroot -p'Huawei@123' logs
mysqladmin: [Warning] Using a password on the command line interface can be insecure.
mysqladmin: Unknown command: 'logs'
[root@localhost ~]# mysqladmin -uroot -p'Huawei@123' flush-logs
mysqladmin: [Warning] Using a password on the command line interface can be insecure.
```

## Step 3 Check whether the MySQL status is normal.

Use the **ping** statement to check whether the MySQL server is normal.

```
mysqladmin -uroot -p'Huawei@123' ping
```

```
[root@localhost ~]# mysqladmin -uroot -p'Huawei@123' ping
mysqladmin: [Warning] Using a password on the command line interface can be insecure.
mysqld is alive
```

#### Step 4 Create and delete a database.

Use the **create** statement to create a database.

```
mysqladmin -uroot -p'Huawei@123' create test
```

Use the **drop** statement to delete a database and its tables.

```
mysqladmin -uroot -p'Huawei@123' drop test
```

```
[root@localhost ~]# mysqladmin -uroot -p'Huawei@123' create test
mysqladmin: [Warning] Using a password on the command line interface can be insecure.
[root@localhost ~]# mysqladmin -uroot -p'Huawei@123' drop test
mysqladmin: [Warning] Using a password on the command line interface can be insecure.
Dropping the database is potentially a very bad thing to do.
Any data stored in the database will be destroyed.

Do you really want to drop the 'test' database [y/N] y
Database "test" dropped
```

## 4.2 Performing Comprehensive MySQL Practices

Create a **Vegetables** database for vegetables in a shopping mall in the following format for subsequent applications to invoke.

ID	Name	Price	Qty	PIC
1	Cabbage		200	Tom
2	Potato	2.60	300	Peter
3	Bok choy	6.00	150	Jim
4	Tomato	5.20	230	Betty
5	Cucumber	8.00	330	Jim

Create user **vegetables\_usser** and allow it to perform SELECT query on the database. In addition, create user **vegetables\_admin** and grant it all permissions on the database.

#### Step 1 Create a database.

Run the following command to create a database:

```
mysql> CREATE DATABASE Vegetables;
```

```
mysql> CREATE DATABASE Vegetables;
Query OK, 1 row affected (0.17 sec)
```

After the creation is complete, view information about the created database.

```
mysql> SHOW databases;
```

```
mysql> SHOW databases;
+-----+
| Database |
+-----+
| Vegetables
| information_schema
| mysql
| performance_schema
| sys
| test
+-----+
6 rows in set (0.01 sec)
```

Use the **USE** statement to access the database, and then use the **SELECT** statement to view information about the database.

```
mysql> USE Vegetables;
mysql> SELECT database();
```

```
mysql> USE Vegetables;
Database changed
mysql> SELECT database();
+-----+
| database() |
+-----+
| Vegetables |
+-----+
1 row in set (0.00 sec)
```

## Step 2 Create a data table.

When creating a table, set **ID** to the primary key and set **Name** and **PIC** to **NOT NULL**.

```
mysql> CREATE TABLE Vegetables ( ID SMALLINT UNSIGNED PRIMARY KEY , Name VARCHAR(10)
NOT NULL , Price DECIMAL(5, 2) , Qty DECIMAL(7, 2) , PIC VARCHAR(10) NOT NULL );
```

To make the statement structure clearer, you can also write it as follows:

```
mysql> CREATE TABLE Vegetables (
-> ID SMALLINT UNSIGNED PRIMARY KEY,
-> Name VARCHAR(10) NOT NULL,
-> Price DECIMAL(5, 2),
-> Qty DECIMAL(7, 2),
-> PIC VARCHAR(10) NOT NULL
-> );
```

```
mysql> CREATE TABLE Vegetables (
-> ID SMALLINT UNSIGNED PRIMARY KEY,
-> Name VARCHAR(10) NOT NULL,
-> Price DECIMAL(5, 2),
-> Qty DECIMAL(7, 2),
-> PIC VARCHAR(10) NOT NULL
-> );
```

After the table is created, use related statements to view information about the table, such as **SHOW** and **SELECT**.

Check whether the table is successfully created.

```
mysql> SHOW TABLES;
+-----+
| Tables_in_Vegetables |
+-----+
| Vegetables           |
+-----+
1 row in set (0.00 sec)
```

View the table structure.

```
mysql> DESC Vegetables;
+-----+-----+-----+-----+-----+
| Field | Type            | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| ID    | smallint unsigned | NO   | PRI | NULL    |       |
| Name  | varchar(10)       | NO   |     | NULL    |       |
| Price | decimal(5,2)      | YES  |     | NULL    |       |
| Qty   | decimal(7,2)      | YES  |     | NULL    |       |
| PIC   | varchar(10)       | NO   |     | NULL    |       |
+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)
```

Check the table status.

```
mysql> SHOW TABLE STATUS LIKE 'Vegetables'
+-----+
| Name      | Engine | Version | Row_format | Rows | Avg_row_length | Data_length | Max_data_length | Index_length | Data_free | Auto_increment | Create_time          | Update_time | Check_time | Collation | Checksum | Create_
e_options | Comment |
+-----+
| Vegetables | InnoDB | 10 | Dynamic | 0 | 0 | 16384 | 0 | 0 | 0 | NULL | 2023-02-24 18:07:23 | NULL | NULL | utf8mb4_0900_ai_ci | NULL |
+-----+
1 row in set (0.00 sec)
```

### Step 3 Add data.

Add data to the table in sequence.

```
mysql> INSERT Vegetables VALUES
-> (1,'Cabbage',NULL,200,'Tom'),
-> (2,'Potato',2.60,300,'Peter'),
-> (3,'Bok choy',6,150,'Jim'),
-> (4,'Tomato',5.2,230,'Betty'),
-> (5,'Cucumber',8,330,'Jim');
```

```
mysql> INSERT Vegetables VALUES
-> (1,'Cabbage',NULL,200,'Tom'),
-> (2,'Potato',2.60,300,'Peter'),
-> (3,'Bok choy',6,150,'Jim'),
-> (4,'Tomato',5.2,230,'Betty'),
-> (5,'Cucumber',8,330,'Jim');
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

After the table is added, view data in the table, as shown in the following figure.

```
mysql> SELECT * FROM Vegetables;
+----+-----+-----+-----+
| ID | Name | Price | Qty | PIC |
+----+-----+-----+-----+
| 1 | Cabbage | NULL | 200.00 | Tom |
| 2 | Potato | 2.60 | 300.00 | Peter |
| 3 | Bok choy | 6.00 | 150.00 | Jim |
| 4 | Tomato | 5.20 | 230.00 | Betty |
| 5 | Cucumber | 8.00 | 330.00 | Jim |
+----+-----+-----+-----+
5 rows in set (0.00 sec)
```

- Question: Why are the values in **Price** and **Qty** automatically saved to two decimal places?

Answer: When the table is created, the **DECIMAL** modifier is added to **Price** and **Qty** to make the values accurate to two decimal places.

#### Step 4 Perform data queries.

Query information about the vegetables managed by Jim from the table.

```
mysql> SELECT * FROM Vegetables WHERE PIC='Jim';
```

```
mysql> SELECT * FROM Vegetables WHERE PIC='Jim';
+----+-----+-----+-----+
| ID | Name | Price | Qty | PIC |
+----+-----+-----+-----+
| 3 | Bok choy | 6.00 | 150.00 | Jim |
| 5 | Cucumber | 8.00 | 330.00 | Jim |
+----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Query the names and prices of the vegetables managed by Jim.

```
mysql> SELECT Name,Price FROM Vegetables WHERE PIC='Jim';
```

```
mysql> SELECT Name,Price FROM Vegetables WHERE PIC='Jim';
+-----+-----+
| Name | Price |
+-----+-----+
| Bok choy | 6.00 |
| Cucumber | 8.00 |
+-----+-----+
2 rows in set (0.00 sec)
```

Query information about the vegetables managed by Jim or Tom from the table.

```
mysql> SELECT * FROM Vegetables WHERE PIC='Jim' or PIC='Tom';
```

```
mysql> SELECT * FROM Vegetables WHERE PIC='Jim' or PIC='Tom';
+----+-----+-----+-----+
| ID | Name   | Price | Qty   | PIC  |
+----+-----+-----+-----+
| 1  | Cabbage | NULL  | 200.00 | Tom  |
| 3  | Bok choy | 6.00  | 150.00 | Jim   |
| 5  | Cucumber | 8.00  | 330.00 | Jim   |
+----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Query information about the vegetables whose prices are higher than 7.00 and quantity is less than 180.

```
mysql> SELECT * FROM Vegetables WHERE Price>7 and Qty<180;
```

```
mysql> SELECT * FROM Vegetables WHERE Price>7 and Qty<180;
Empty set (0.00 sec)
```

Query information about the vegetables managed by the salesmen whose names start with T (fuzzy query).

```
mysql> SELECT * FROM Vegetables WHERE PIC LIKE 'T%';
```

```
mysql> SELECT * FROM Vegetables WHERE PIC LIKE 'T%';
+----+-----+-----+-----+
| ID | Name   | Price | Qty   | PIC  |
+----+-----+-----+-----+
| 1  | Cabbage | NULL  | 200.00 | Tom  |
+----+-----+-----+-----+
1 row in set (0.00 sec)
```

## Step 5 Modify data.

Change the price of the vegetable whose ID is 1 to 4.2.

```
mysql> UPDATE Vegetables SET Price=4.2 WHERE ID=1;
```

```
mysql> UPDATE Vegetables SET Price=4.2 WHERE ID=1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Check whether the modification takes effect.

```
mysql> SELECT Price FROM Vegetables WHERE ID=1;
+-----+
| Price |
+-----+
| 4.20  |
+-----+
1 row in set (0.00 sec)
```

## Step 6 Create users and grant permissions.

Create users **vegetable\_user** and **vegetable\_admin** as planned and set their passwords to **Huawei@123**.

```
mysql> CREATE USER vegetable_user@localhost identified by 'Huawei@123';
```

```
Query OK, 0 rows affected (0.50 sec)

mysql> CREATE USER vegetable_admin@localhost identified by 'Huawei@123';
Query OK, 0 rows affected (0.01 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.41 sec)
```

```
mysql> CREATE USER vegetable_user@localhost identified by 'Huawei@123';
Query OK, 0 rows affected (0.50 sec)

mysql> CREATE USER vegetable_admin@localhost identified by 'Huawei@123';
Query OK, 0 rows affected (0.01 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.41 sec)
```

- Question: Are there any deficiencies in the users created by running the preceding commands? How do you make up for such deficiencies?

Answer: The users created by running the preceding commands can log in to the database only on the local host but cannot log in to the database over the Internet. When creating a user, you can specify the hosts or network segments that are allowed to log in to the MySQL database. For example, to allow the **192.168.1.0** network segment, use the following statement:

```
CREATE USER vegetable_user@'192.168.1.%' identified by 'Huawei@123';
```

To allow all network segments, use the following statement:

```
CREATE USER vegetable_user@'%' identified by 'Huawei@123';
```

Alternatively, use the following statement:

```
CREATE USER vegetable_user@ identified by 'Huawei@123';
```

Modify the attributes of the users.

```
mysql> use mysql;
mysql> UPDATE user SET HOST='%' WHERE user LIKE 'vege%';
```

```
mysql> use mysql;
Database changed
mysql> UPDATE user SET HOST='%' WHERE user LIKE 'vege%';
Query OK, 2 rows affected (0.00 sec)
Rows matched: 2  Changed: 2  Warnings: 0
```

Grant permissions to the created users as planned.

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION;
mysql> flush privileges;
mysql> GRANT ALL PRIVILEGES ON Vegetables.* TO vegetable_admin@'%';
mysql> GRANT SELECT ON Vegetables.* TO vegetable_user@'%';
```

```
mysql> GRANT ALL PRIVILEGES ON Vegetables.* TO vegetable_admin@'%';
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT SELECT ON Vegetables.* TO vegetable_user@'%';
Query OK, 0 rows affected (0.01 sec)
```

Log out of user **root**, log in to the database as users **vegetable\_admin** and **vegetable\_user**, and check whether the configurations take effect. Log in to the database as user **vegetable\_user**.

```
[root@localhost ~]# mysql -uvegetable_user -p'Huawei@123'
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 20
Server version: 8.0.35 Source distribution

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> USE Vegetables;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SELECT * FROM Vegetables;
+----+-----+-----+-----+-----+
| ID | Name   | Price | Qty   | PIC   |
+----+-----+-----+-----+-----+
| 1  | Cabbage | NULL  | 200.00 | Tom   |
| 2  | Potato  | 2.60  | 300.00 | Peter |
| 3  | Bok choy | 6.00  | 150.00 | Jim   |
| 4  | Tomato  | 5.20  | 230.00 | Betty |
| 5  | Cucumber | 8.00  | 330.00 | Jim   |
+----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> UPDATE Vegetables SET Price=5 WHERE ID=1;
ERROR 1142 (42000): UPDATE command denied to user 'vegetable_user'@'localhost' for table 'Vegetables'
```

Log in to the database as user **vegetable\_admin**.

```
[root@localhost ~]# mysql -uvegetable admin -p'Huawei@123'
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 21
Server version: 8.0.35 Source distribution

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> USE Vegetables;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SELECT * FROM Vegetables;
+----+-----+-----+-----+
| ID | Name   | Price | Qty   | PIC   |
+----+-----+-----+-----+
| 1  | Cabbage | NULL  | 200.00 | Tom   |
| 2  | Potato   | 2.60  | 300.00 | Peter |
| 3  | Bok choy | 6.00  | 150.00 | Jim   |
| 4  | Tomato   | 5.20  | 230.00 | Betty |
| 5  | Cucumber | 8.00  | 330.00 | Jim   |
+----+-----+-----+-----+
5 rows in set (0.00 sec)
```

## Step 7 Modify and delete the database, data tables, and users.

Change the table name from **Vegetables** to **Vegetable1**.

```
mysql> ALTER TABLE Vegetables rename as Vegetable1;
```

```
mysql> ALTER TABLE Vegetables rename as Vegetable1;
Query OK, 0 rows affected (0.01 sec)

mysql> SHOW TABLES;
+-----+
| Tables_in_Vegetables |
+-----+
| Vegetable1           |
+-----+
1 row in set (0.01 sec)
```

Delete the **Vegetable1** table.

```
mysql> DROP TABLE Vegetable1;
```

```
mysql> DROP TABLE Vegetable1;
Query OK, 0 rows affected (0.07 sec)

mysql> SHOW TABLES;
Empty set (0.00 sec)
```

Delete the **Vegetables** database.

```
mysql> DROP DATABASE Vegetables;
```

```
mysql> DROP DATABASE Vegetables;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW databases;
+-----+
| Database      |
+-----+
| information_schema |
+-----+
1 row in set (0.01 sec)
```

Log in to the MySQL database as user **root** and delete users **vegetable\_user** and **vegetable\_admin**.

```
mysql> DROP USER vegetable_user, vegetable_admin;
```

```
mysql> DROP USER vegetable_user, vegetable_admin;
Query OK, 0 rows affected (0.00 sec)

mysql> select User FROM user where User like 'vege%';
Empty set (0.00 sec)
```

# 5 LAMP Practices

## 5.1 Introduction

This lab uses **Server01** and **Server04** to set up a WordPress server developed in PHP.

## 5.2 Interconnecting Components in the Early Stage

### 5.2.1 Interconnecting Apache with PHP

Step 1    Install PHP.

On the Apache host, run the following command to install PHP:

```
yum install -y php
```

```
[root@apache-http ~]# yum install -y php
Last metadata expiration check: 0:00:00 ago on Mon Nov 21 17:44:09 2022.
Dependencies resolved.
=====
 Package                                Architecture
=====
Installing:
  php                                     x86_64
Installing dependencies:
  libargon2                               x86_64
  nginx-filesystem                         noarch
  php-cli                                 x86_64
  php-common                             x86_64
Installing weak dependencies:
  php-fpm                                x86_64
=====
Transaction Summary
=====
Install 6 Packages
```

After the installation is complete, run the following command to check whether the installation is correct:

```
php -v
```

```
[root@apache-http ~]# php -v
PHP 8.0.26 (cli) (built: Nov 22 2022 17:44:09) ( NTS )
Copyright (c) The PHP Group
Zend Engine v4.0.26, Copyright (c) Zend Technologies
```

Step 2    Modify Apache configurations.

Add the PHP configuration to the Apache main configuration file as follows.

```
<IfModule mime_module>
#
# TypesConfig points to the file containing th
# filename extension to MIME-type.
#
TypesConfig /etc/mime.types

#
# AddType allows you to add to or override the
# file specified in TypesConfig for specific f
#
#AddType application/x-gzip .tgz
#
# AddEncoding allows you to have certain brows
# information on the fly. Note: Not all browse
#
#AddEncoding x-compress .Z
#AddEncoding x-gzip .gz .tgz
#
# If the AddEncoding directives above are comm
# probably should define those extensions to i
#
AddType application/x-compress .Z
AddType application/x-gzip .gz .tgz
AddType application/x-httpd-php .php
```

To prevent interference, back up or delete all configurations in the previous labs. For example, copy all sub-configuration files in **conf.d** to the new **conf.bk** directory and retain only the **php.conf** file related to PHP.

```
cd /etc/httpd/conf.d
mkdir conf.bk && mv * conf.bk/
mv conf.bk/php.conf .
```

After the configuration is complete, the files in the **conf.d** directory are as follows.

```
[root@apache-http conf.d]# ll
total 8.0K
drwxr-xr-x 2 root root 4.0K [REDACTED] conf.bk
-rw-r--r-- 1 root root 1.6K [REDACTED] php.conf
[root@apache-http conf.d]#
```

Restart the httpd service and check whether it can be accessed.

### Step 3 Verify the configurations.

Create **index.php** in the root directory of httpd to test whether Apache and PHP can work properly. The content of **index.php** is as follows:

```
<?php
phpinfo();
?>
```

In the address box of the browser, enter *Apache\_host\_EIP/index.php* to access the Apache server. If the following page is displayed, the configuration is correct.

System	Linux apache-http 5.10.0-60.18.0.50.oe2203.x86_64 #1 SMP Wed Mar 30 03:12:24 UTC 2022 x86_64
Build Date	Nov 22 2022 17:44:09
Build System	Linux obs-worker1639015616-x86-0004 4.19.90-2003.4.0.0036.oe1.x86_64 #1 SMP Mon Mar 23 19:10:41 UTC 2020 x86_64 x86_64 GNU/Linux
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	/etc/php.d/20-bz2.ini, /etc/php.d/20-calendar.ini, /etc/php.d/20-ctype.ini, /etc/php.d/20-curl.ini, /etc/php.d/20-exif.ini, /etc/php.d/20-fileinfo.ini, /etc/php.d/20-ftp.ini, /etc/php.d/20-gettext.ini, /etc/php.d/20-iconv.ini, /etc/php.d/20-phar.ini, /etc/php.d/20-sockets.ini, /etc/php.d/20-tokenizer.ini
PHP API	20200930
PHP Extension	20200930
Zend Extension	420200930

If the page is not displayed, check whether the php-fpm service is normal, as shown in the following figure.

```
[root@apache-http html]# systemctl start php-fpm
[root@apache-http html]# systemctl status php-fpm
● php-fpm.service - The PHP FastCGI Process Manager
   Loaded: loaded (/usr/lib/systemd/system/php-fpm.service; disabled; vendor preset: disabled)
   Active: active (running) since Wed 2023-03-01 15:25:04 CST; 1s ago
     Main PID: 2262 (php-fpm)
       Status: "Ready to handle connections"
         Tasks: 6 (limit: 21420)
        Memory: 6.2M
      CGroup: /system.slice/php-fpm.service
```

## 5.2.2 Interconnecting PHP with MySQL

### Step 1    Grant the MySQL access permission.

This lab uses user **root** for testing. Check whether user **root** has the required permissions, as shown in the following figure.

```
mysql> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SELECT Host FROM user WHERE User='root';
+-----+
| Host |
+-----+
| %    |
+-----+
1 row in set (0.00 sec)
```

If user **root** is not allowed to access all other hosts or the Apache host, complete the configuration by referring to the MySQL section.

After checking the permissions of user **root**, run the following commands on the Apache host to install and connect to the MySQL server:

```
yum install -y mysql
mysql -h'192.168.1.14' -uroot -p'Huawei@123'
```

```
[root@apache-http html]# mysql -h'192.168.1.14' -uroot -p'Huawei@123'  
mysql: [Warning] Using a password on the command line interface can be insecure.  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 16  
Server version: 8.0.28 Source distribution  
  
Copyright (c) 2000, 2022, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
mysql> exit  
Bye
```

## Step 2 Install the driver for connecting PHP to MySQL.

Run the following command to install the driver:

```
yum install -y php-mysqlnd
```

```
[root@apache-http html]# yum install -y php-mysqlnd  
Last metadata expiration check: 0:34:45 ago on [REDACTED] PM CST.  
Dependencies resolved.  
=====  
           Package                                Architecture  
=====  
Installing:  
  php-mysqlnd                                     x86_64  
Installing dependencies:  
  php-pdo                                         x86_64  
  
Transaction Summary  
=====  
Install  2 Packages
```

## Step 3 Compile a test file for connecting PHP to MySQL.

Create a **conn\_mysql.php** file in the root directory of httpd and enter the following content:

```
<?php  
$con = mysqli_connect("192.168.1.14","root","Huawei@123");  
if ($con)  
    echo 'OK';  
else  
    echo 'NOT OK';  
$con->close();  
?>
```

Enter *Apache\_host\_EIP/conn\_mysql.php* in the address box of the browser to access the Apache server and check whether OK is returned.



## 5.3 Performing LAMP Practices

### 5.3.1 Introduction

WordPress is an open source blog platform developed using PHP. Due to its ease of use and openness, WordPress is widely used in various industries. This LAMP lab will be completed based on WordPress.

### 5.3.2 Preparing Resources

Step 1 Download and decompress the WordPress installation package.

Run the following commands on the Apache host to download the WordPress installation package:

```
cd /home  
mkdir wordpress  
wget https://wordpress.org/latest.tar.gz
```

```
[root@apache-http html]# cd /home  
[root@apache-http home]# mkdir wordpress  
[root@apache-http home]# wget https://wordpress.org/latest.tar.gz  
--2023-03-01 16:09:11-- https://wordpress.org/latest.tar.gz  
Resolving wordpress.org (wordpress.org)... 198.143.164.252  
Connecting to wordpress.org (wordpress.org)|198.143.164.252|:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 22751086 (22M) [application/octet-stream]  
Saving to: 'latest.tar.gz'  
  
latest.tar.gz                                              100%[=====]  
  
2023-03-01 16:09:19 (3.01 MB/s) - 'latest.tar.gz' saved [22751086/22751086]
```

Run the following commands to decompress the downloaded package:

```
mv latest.tar.gz wordpress/  
cd wordpress/  
tar -xzf latest.tar.gz
```

```
[root@apache-http home]# mv latest.tar.gz wordpress/  
[root@apache-http home]# cd wordpress/  
[root@apache-http wordpress]# tar -xzf latest.tar.gz  
[root@apache-http wordpress]# ls  
latest.tar.gz  wordpress
```

If the **tar** command is not available, install it using the Yum repository.

## Step 2 Create the database required by WordPress.

Create a **wordpress** database required by WordPress on the MySQL server.

```
mysql> CREATE DATABASE wordpress;
```

```
mysql> CREATE DATABASE wordpress;
Query OK, 1 row affected (0.00 sec)
```

Create a dedicated user named **wp** for the database and grant the read and write permissions on the database to the user.

```
mysql> CREATE USER wp@'%' identified by 'Huawei@123';
mysql> GRANT ALL PRIVILEGES ON wordpress.* TO 'wp'@'%';
mysql> FLUSH PRIVILEGES;
```

```
mysql> CREATE DATABASE wordpress;
Query OK, 1 row affected (0.00 sec)

mysql> CREATE USER wp@'%' identified by 'Huawei@123';
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT ALL PRIVILEGES ON wordpress.* TO 'wp'@'%';
Query OK, 0 rows affected (0.01 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)
```

After the configuration is complete, check whether you can access the database as user **wp** on the Apache host.

```
[root@apache-http wordpress]# mysql -h'192.168.38.167' -uwp -p'Huawei@123'
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14
Server version: 8.0.28 Source distribution

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SHOW databases;
+-----+
| Database      |
+-----+
| information schema |
| wordpress      |
+-----+
2 rows in set (0.00 sec)
```

### 5.3.3 Installing and Testing WordPress

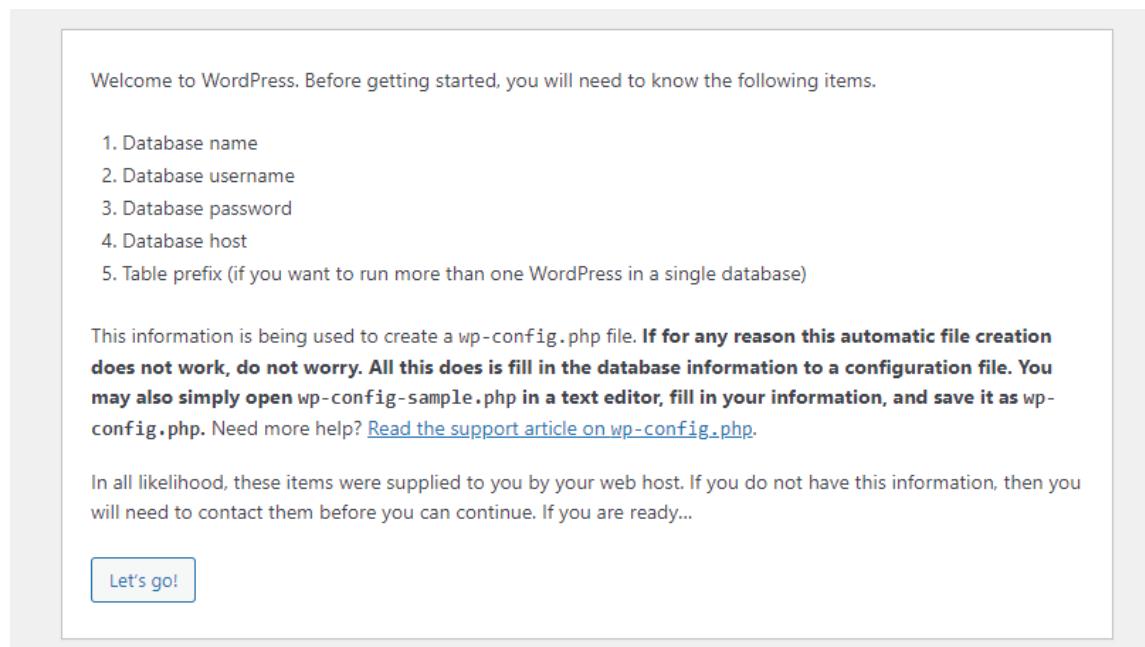
Run the following commands to copy all files in the decompressed WordPress directory to the root directory of the httpd service:

```
cd /home/wordpress/wordpress/
cp -r * /var/www/html
ls /var/www/html
```

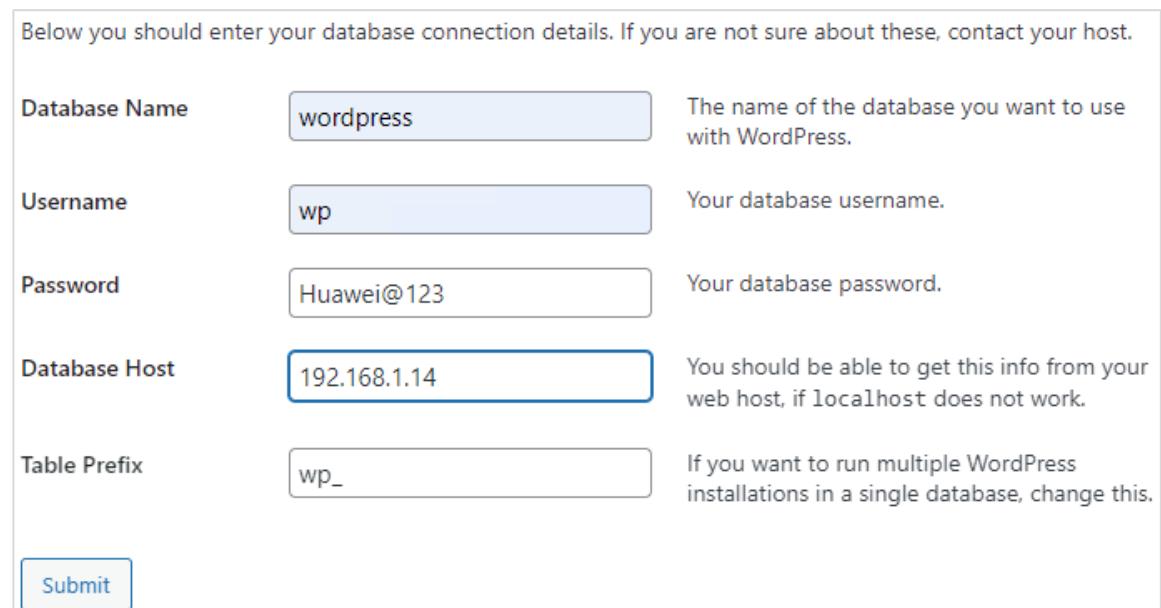
```
[root@apache http]# cd /home/wordpress/wordpress/
[root@apache http wordpress]# cp -r * /var/www/html
[root@apache http wordpress]# ls /var/www/html
index.php      readme.html      wp-admin      wp-comments-post.php  wp-content      wp-includes      wp-load.php      wp-mail.php      wp-signup.php      xmlrpc.php
license.txt    wp-activate.php   wp-blog-header.php  wp-config-sample.php  wp-cron.php    wp-links-opml.php  wp-login.php    wp-settings.php  wp-trackback.php
```

- Note: During the copy, the system asks you whether to overwrite the original **index.php** file. Enter **y** to overwrite the original **index.php** file. Alternatively, back up the original **index.php** file and copy the file again.

After the copy is complete, enter **http://Apache\_server\_EIP/wp-admin/install.php** in the address box of the browser to start installation, as shown in the following figure.



Click **Let's go** to start installation. On the next page, enter the MySQL database information, as shown in the following figure.



Below you should enter your database connection details. If you are not sure about these, contact your host.

Database Name	<input type="text" value="wordpress"/>	The name of the database you want to use with WordPress.
Username	<input type="text" value="wp"/>	Your database username.
Password	<input type="text" value="Huawei@123"/>	Your database password.
Database Host	<input type="text" value="192.168.1.14"/>	You should be able to get this info from your web host, if localhost does not work.
Table Prefix	<input type="text" value="wp_"/>	If you want to run multiple WordPress installations in a single database, change this.

**Submit**

Click **Submit**. A message is displayed, indicating that the WordPress configuration file **wp-config.php** cannot be found and needs to be manually created, as shown in the following figure.

Unable to write to wp-config.php file.  
You can create the wp-config.php file manually and paste the following text into it.

Configuration rules for wp-config.php:

```
/* Add any custom values between this line and the "stop editing" line. */  
  
/* That's all, stop editing! Happy publishing. */  
  
/** Absolute path to the WordPress directory. */  
if ( ! defined( 'ABSPATH' ) ) {  
    define( 'ABSPATH', __DIR__ . '/' );  
}  
  
/** Sets up WordPress vars and included files. */  
require_once ABSPATH . 'wp-settings.php';
```

After you've done that, click "Run the installation".

[Run the installation](#)

Manually create **wp-config.php** in the **/var/www/html** directory, copy the content highlighted in the red box in the preceding figure to the file, and click **Run the installation**. The installation program automatically switches to the configuration page. The following figure shows the parameters that have been set.

## Information needed

Please provide the following information. Do not worry, you can always change these settings later.

Site Title

Welcome openEuler

Username

openEuler

Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.

Password

Huawei@123

 Hide

Medium

**Important:** You will need this password to log in. Please store it in a secure location.

Your Email

test@test.com

Double-check your email address before continuing.

Search engine visibility

Discourage search engines from indexing this site

It is up to search engines to honor this request.

[Install WordPress](#)

After the configuration is complete, click **Install WordPress**. After the installation is complete, the following page is displayed.

Success!

WordPress has been installed. Thank you, and enjoy!

Username

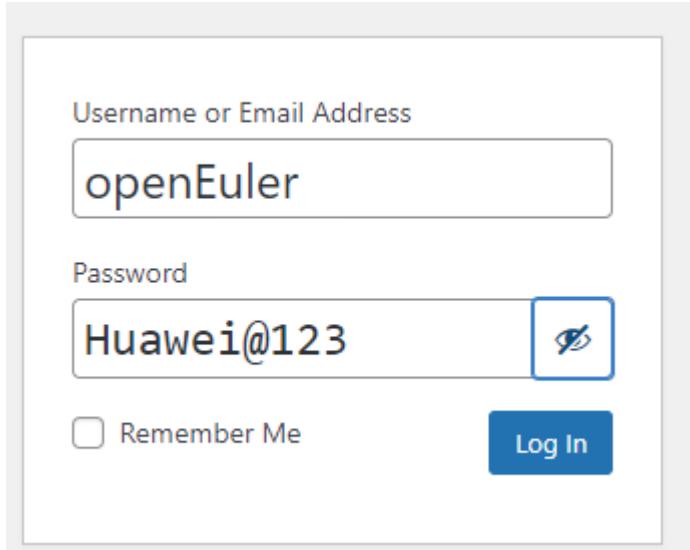
openEuler

Password

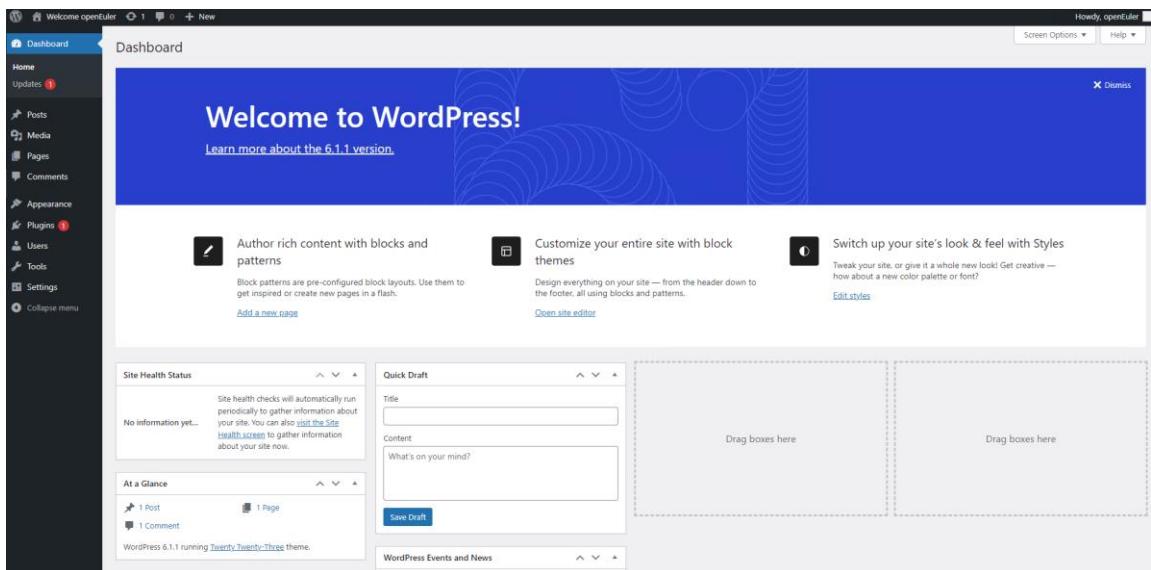
Your chosen password.

[Log In](#)

Click **Log In** and enter the user name and password to log in, as shown in the following figure.



Click **Log In** to log in to the home page, as shown in the following figure.



- Question: What are L, A, M, and P in WordPress practice?

Answer:

In practice, both the WordPress installation package, Apache, and MySQL run on the openEuler OS. openEuler is an advanced Linux distribution, which is the embodiment of "L".

The WordPress installation and access after the installation are implemented through web services. In this practice, web services are provided by Apache, which is the embodiment of "A".

Before installing WordPress, you have created the **wordpress** database in MySQL. During the installation, you have entered related information, such as the user name and password. After the installation is complete, WordPress automatically creates tables and data in the database, as shown in the following figure.

```
mysql> SHOW TABLES;
+-----+
| Tables_in_wordpress |
+-----+
| wp_commentmeta
| wp_comments
| wp_links
| wp_options
| wp_postmeta
| wp_posts
| wp_term_relationships
| wp_term_taxonomy
| wp_termmeta
| wp_terms
| wp_usermeta
| wp_users
+-----+
12 rows in set (0.00 sec)
```

These are the embodiments of "M".

WordPress is developed based on PHP, which is the embodiment of "P".

-----End-----

Huawei openEuler Certification Training

# HCIP-openEuler

## Lab Guide

Issue: 1.0



Huawei Technologies Co., Ltd.

**Copyright © Huawei Technologies Co., Ltd. 2024. All rights reserved.**

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

### **Trademarks and Permissions**



HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

### **Notice**

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base Bantian, Longgang Shenzhen 518129 People's Republic of China

Website: <https://e.huawei.com>

## Huawei Certification System

Huawei Certification is an integral part of the company's Platform + Ecosystem strategy. It supports the development of ICT infrastructure that features Cloud-Pipe-Device synergy. Our certification is always evolving to reflect the latest trends in ICT development. Huawei Certification consists of three categories: ICT Infrastructure Certification, Basic Software & Hardware Certification, and Cloud Platform & Services Certification, making it the most extensive technical certification program in the industry.

Huawei offers three levels of certification: Huawei Certified ICT Associate (HCIA), Huawei Certified ICT Professional (HCIP), and Huawei Certified ICT Expert (HCIE).

Our programs cover all ICT fields and follow the industry's trend of ICT convergence. With our leading talent development system and certification standards, we are committed to fostering new digital ICT talent and building a sound ICT talent ecosystem.

HCIP-openEuler is intended for frontline engineers at Huawei regional offices or representative offices, and other personnel who want to learn openEuler O&M technologies. HCIP-openEuler certification covers common openEuler enterprise service management, openEuler HA cluster architecture, openEuler storage management, openEuler automated O&M, Linux shell scripts, openEuler system security hardening, and openEuler system monitoring.

Huawei certification helps you unlock opportunities to advance your career and take one more step towards the top of the industry.

## Huawei Career Certification

Cloud Platform & Services		Cloud Computing	Cloud Service	Big Data	GaussDB	IoT
Basic Software & Hardware	Basic Software & Hardware Domain	openEuler	openGauss	HarmonyOS	AI	Kunpeng
IT Infrastructure	IT Infrastructure Domain	Storage	Collaboration	Intelligent Vision	Digital Power	MDC
ICT Infrastructure	Datacom Domain	Datacom	WLAN		Security	Huawei Certified ICT Professional
	Optical Domain	Transmission		Access		HCI
	Wireless Domain	5G		LTE		HCI Associate
Huawei Certified ICT Expert		HCIE				
Huawei Certified ICT Associate		HCIA				

# About This Document

---

## Overview

This document is intended for trainees preparing for the HCIP-openEuler certification exam and those who are interested in building enterprise services and shell scripts or performing automated O&M using Zabbix or Salt on openEuler and other Linux distributions.

## Description

This lab guide consists of five labs, starting from basic device configurations, and describes how to install and configure cluster software on openEuler, including the Linux Virtual Server (LVS), NGINX, HAProxy, and Keepalived.

- Lab 1: LVS configuration
  - This lab employs the LVS-NAT and LVS-DR modes to balance the loads of web servers.
- Lab 2: NGINX reverse proxy and load balancing configuration
  - This lab mainly introduces load balancing algorithms.
- Lab 3: HAProxy configuration
  - This lab focuses on how to use the monitoring function of HAProxy, configure HAProxy logs, and configure the access control feature of HAProxy.
- Lab 4: Keepalived configuration
  - This lab uses Keepalived to provide high availability (HA) for NGINX and implement an NGINX cluster together with LVS.
- Lab 5: Redis operations
  - This lab involves addition, deletion, query, and modification operations and applies Redis in WordPress.

## Background Knowledge Required

This course is for Huawei's basic certification. To better understand this course, the intended audience should:

- Have basic Linux knowledge. You are advised to complete HCIA-openEuler learning and pass the HCIA-openEuler certification exam.

# Lab Environment Preparation

## Checking Devices

Before starting the labs, each group of trainees should apply for ECSs on Huawei Cloud according to the following table.

Device Name	Specifications	Remarks
ECS	1 vCPUs   1 GiB   s7.small.1	The required quantity is subject to each lab.

# Contents

---

<b>1 LVS Configuration .....</b>	<b>1</b>
1.1 LVS-NAT Configuration.....	1
1.1.1 Introduction .....	1
1.1.2 Networking.....	1
1.1.3 Procedure.....	2
1.2 LVS-DR Configuration.....	5
1.2.1 Introduction .....	5
1.2.2 Networking.....	5
1.2.3 Procedure.....	6
<b>2 NGINX Reverse Proxy and Load Balancing Configuration .....</b>	<b>9</b>
2.1 Introduction .....	9
2.2 Networking.....	9
2.3 Procedure .....	10
<b>3 HAProxy Configuration .....</b>	<b>16</b>
3.1 Introduction .....	16
3.2 Networking.....	16
3.3 Procedure .....	17
3.3.1 Implementing Basic Load Balancing with HAProxy .....	17
3.3.2 Configuring GUI-based HAProxy Monitoring .....	18
3.3.3 Configuring HAProxy Logs.....	19
3.3.4 Configuring HAProxy ACLs.....	20
<b>4 Keepalived Configuration.....</b>	<b>22</b>
4.1 NGINX HA Cluster Configuration with Keepalived .....	22
4.1.1 Introduction .....	22
4.1.2 Networking.....	22
4.1.3 Procedure .....	23
4.2 NGINX Cluster Configuration with Keepalived + LVS .....	27
4.2.1 Introduction .....	27
4.2.2 Networking.....	27
4.2.3 Procedure .....	27
<b>5 Basic Redis Operations.....</b>	<b>31</b>
5.1 Introduction .....	31
5.2 Performing Basic Redis Operations .....	31
5.3 Configuring Redis as the Cache of WordPress .....	35

5.3.1 Introduction .....	35
5.3.2 Networking.....	35
5.3.3 Procedure.....	35

# 1

# LVS Configuration

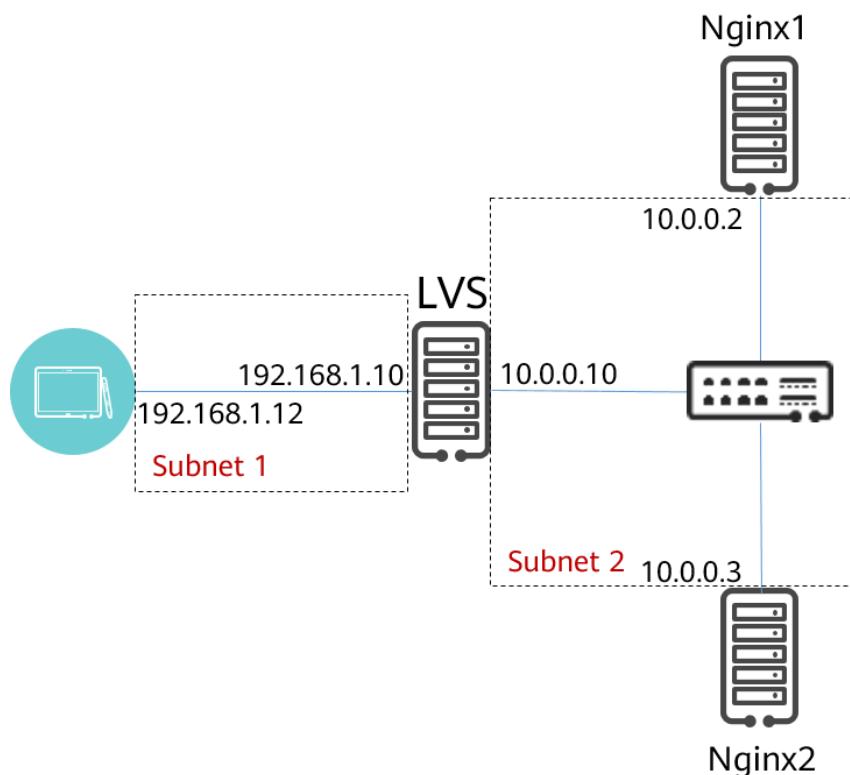
## 1.1 LVS-NAT Configuration

### 1.1.1 Introduction

This lab shows how to access an NGINX cluster in LVS-NAT mode. In actual scenarios, the system returns the same content when the cluster provides the service externally. In this lab, to better explain LVS algorithms, the NGINX service displays different contents. For example, "hello,10.0.0.2" is displayed for Nginx1, and "hello,10.0.0.3" is displayed for Nginx2.

### 1.1.2 Networking

This lab requires four ECSs that functions as the client, LVS, Nginx1, and Nginx2. The virtual IP address (VIP) of the LVS is 192.168.1.10/24, and its director IP address (DIP) is 10.0.0.10/24. The real IP address (RIP) of Nginx1 is 10.0.0.2/24, and the RIP of Nginx2 is 10.0.0.3/24. The RIPS are accessed by the client, as shown in the following figure.



On Huawei Cloud, subnet 1 and subnet 2 are two subnets in the same virtual private cloud.

### 1.1.3 Procedure

#### Step 1 Configure Nginx1 and Nginx2.

**Set the gateway of Nginx1 and Nginx2 to the LVS DIP** according to the preceding figure. Command example:

```
nmcli con mod ens224 ipv4.address 10.0.0.2/24 ipv4.gateway 10.0.0.10  
nmcli con down ens224 & nmcli con up ens224
```

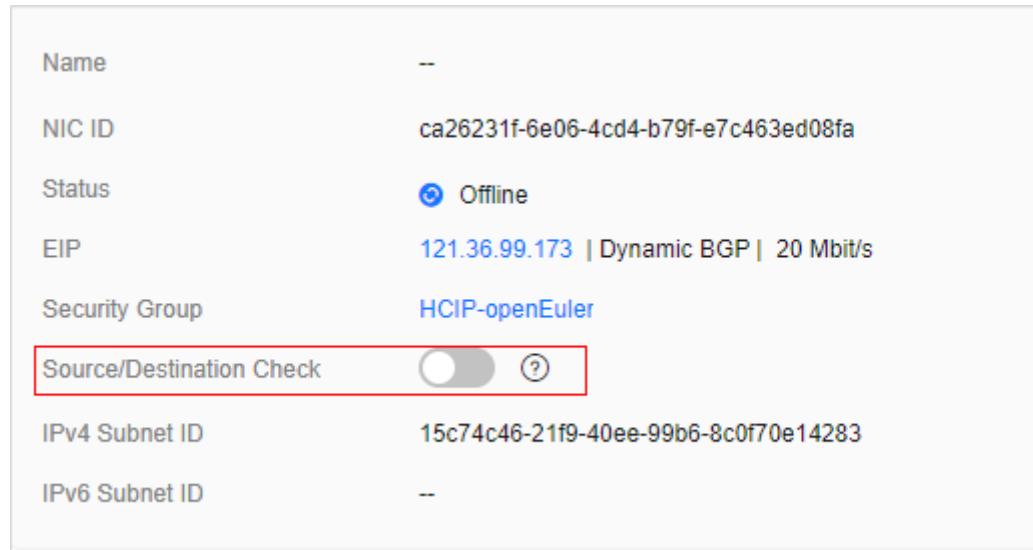
Note: After the modification is done, the network will be interrupted. Log in to the ECS using VNC and perform following operations.

Configure Nginx1 and Nginx2 based on the knowledge in course 1. It is required that the system returns "hello,10.0.0.2" when port 80 of Nginx1 is accessed and "hello,10.0.0.3" port 80 of Nginx2 is accessed. Example:

```
[root@Nginx1 ~]# curl 10.0.0.2  
"hello,10.0.0.2"  
[root@Nginx2 ~]# curl 10.0.0.3  
"hello,10.0.0.3"
```

#### Step 2 Preconfigure the LVS.

Configure two NICs for the LVS ECS, one for the VIP and the other for the DIP. The DIP is an intranet IP address and does not need a gateway. The VIP is a public IP address and needs to be configured with gateway information. In addition, to ensure normal IP forwarding, disable the **Source/Destination Check** item, as shown in the following figure.



Enable the IP forwarding function on the LVS ECS.

```
[root@Cluster1 ~]# sed -i "s/ip_forward=0/ip_forward=1/g" /etc/sysctl.conf
```

```
[root@Cluster1 ~]# sysctl -p | grep net.ipv4.ip_forward  
net.ipv4.ip_forward = 1  
[root@Cluster1 ~]# sysctl -a | grep net.ipv4.ip_forward  
net.ipv4.ip_forward = 1
```

After the configuration, test whether Nginx1 and Nginx2 can be accessed.

```
[root@Cluster1 ~]# curl 10.0.0.2  
"hello,10.0.0.2"  
[root@Cluster1 ~]# curl 10.0.0.3  
"hello,10.0.0.3"
```

### Step 3 Install and configure the LVS.

On the LVS, run the following command to install ipvsadm:

```
[root@Cluster1 ~]# yum install -y ipvsadm
```

After the installation is complete, create a configuration file required for starting ipvsadm.

```
[root@Cluster1 ~]# touch /etc/sysconfig/ipvsadm
```

Start the service and check whether the service is started normally.

```
[root@Cluster1 ~]# systemctl enable ipvsadm --now  
[root@Cluster1 ~]# systemctl status ipvsadm  
● ipvsadm.service - Initialise the Linux Virtual Server  
   Loaded: loaded (/usr/lib/systemd/system/ipvsadm.service; enabled; vendor preset: disabled)  
   Active: active (exited) since Mon 2023-04-10 08:07:56 UTC; 2s ago  
     Process: 9560 ExecStart=/bin/bash -c exec /sbin/ipvsadm-restore </etc/sysconfig/ipvsadm (code=exited, status=0/SUCCESS)  
    Main PID: 9560 (code=exited, status=0/SUCCESS)  
Apr 10 08:07:56 cluster1 systemd[1]: Starting Initialise the Linux Virtual Server...  
Apr 10 08:07:56 cluster1 systemd[1]: Finished Initialise the Linux Virtual Server.
```

If the service is started properly, run the following command to create a cluster using the round robin algorithm:

```
[root@Cluster1 ~]# ipvsadm -A -t 192.168.1.10:80 -s rr
```

Add Nginx1 and Nginx2 as backend real servers, or RSs.

```
[root@Cluster1 ~]# ipvsadm -a -t 192.168.1.10:80 -r 10.0.0.2 -m  
[root@Cluster1 ~]# ipvsadm -a -t 192.168.1.10:80 -r 10.0.0.3 -m
```

Check whether the configuration takes effect.

```
[root@Cluster1 ~]# ipvsadm -Ln  
IP Virtual Server version 1.2.1 (size=4096)  
Prot LocalAddress:Port Scheduler Flags  
-> RemoteAddress:Port           Forward Weight ActiveConn InActConn  
TCP  192.168.1.10:80  rr  
  -> 10.0.0.2:80                Masq    1      0      0  
  -> 10.0.0.3:80                Masq    1      0      0
```

```
[root@Cluster1 ~]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
    -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  192.168.1.10:80 rr
    -> 10.0.0.2:80                 Masq    1      0      0
    -> 10.0.0.3:80                 Masq    1      0      0
```

#### Step 4 Check the LVS configuration.

On the ECS where the LVS resides, use the LVS VIP to repeatedly access the web service. The following information is displayed:

```
[root@Client ~]# curl 192.168.1.10
"hello,10.0.0.2"
[root@Client ~]# curl 192.168.1.10
"hello,10.0.0.3"
[root@Client ~]# curl 192.168.1.10
"hello,10.0.0.2"
[root@Client ~]# curl 192.168.1.10
"hello,10.0.0.3"
```

Or, use different browsers on the local PC to repeatedly access the EIP of the ECS and view the responses.

It shows that the LVS forwards the requests to Nginx1 and Nginx2 in round robin mode and returns the corresponding contents to the client.

- Question: Is it mandatory to set the gateway of Nginx1 and Nginx2 to the LVS DIP?  
Answer: Yes.
- Task: Change the LVS algorithm to weighted round robin.

#### Step 1 Modify the LVS configuration.

Run the following command to change the LVS algorithm to weighted round robin:

```
[root@Cluster1 ~]# ipvsadm -E -t 192.168.1.10:80 -s wrr
```

Check whether the modification takes effect.

```
[root@Cluster1 ~]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
    -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  192.168.1.10:80 wrr
    -> 10.0.0.2:80                 Masq    1      0      0
    -> 10.0.0.3:80                 Masq    1      0      0
```

#### Step 2 Change the weights of the two RSs.

Change the weight of Nginx1 to 2.

```
[root@Cluster1 ~]# ipvsadm -e -t 192.168.1.10:80 -r 10.0.0.2 -m -w 2
```

Check whether the modification takes effect.

```
[root@Cluster1 ~]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
    -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP   192.168.1.10:80 wrr
      -> 10.0.0.2:80                 Masq    2        0        0
      -> 10.0.0.3:80                 Masq    1        0        0
```

### Step 3 Check the LVS configuration.

Log in to the client ECS, use the LVS VIP to repeatedly access the web service. The following information is displayed:

```
[root@Cluster1 ~]# curl 192.168.1.10
"hello,10.0.0.3"
[root@Cluster1 ~]# curl 192.168.1.10
"hello,10.0.0.2"
[root@Cluster1 ~]# curl 192.168.1.10
"hello,10.0.0.2"
[root@Cluster1 ~]# curl 192.168.1.10
"hello,10.0.0.3"
[root@Cluster1 ~]# curl 192.168.1.10
"hello,10.0.0.2"
```

As shown in the preceding figure, the LVS algorithm has been changed to weighted round robin.

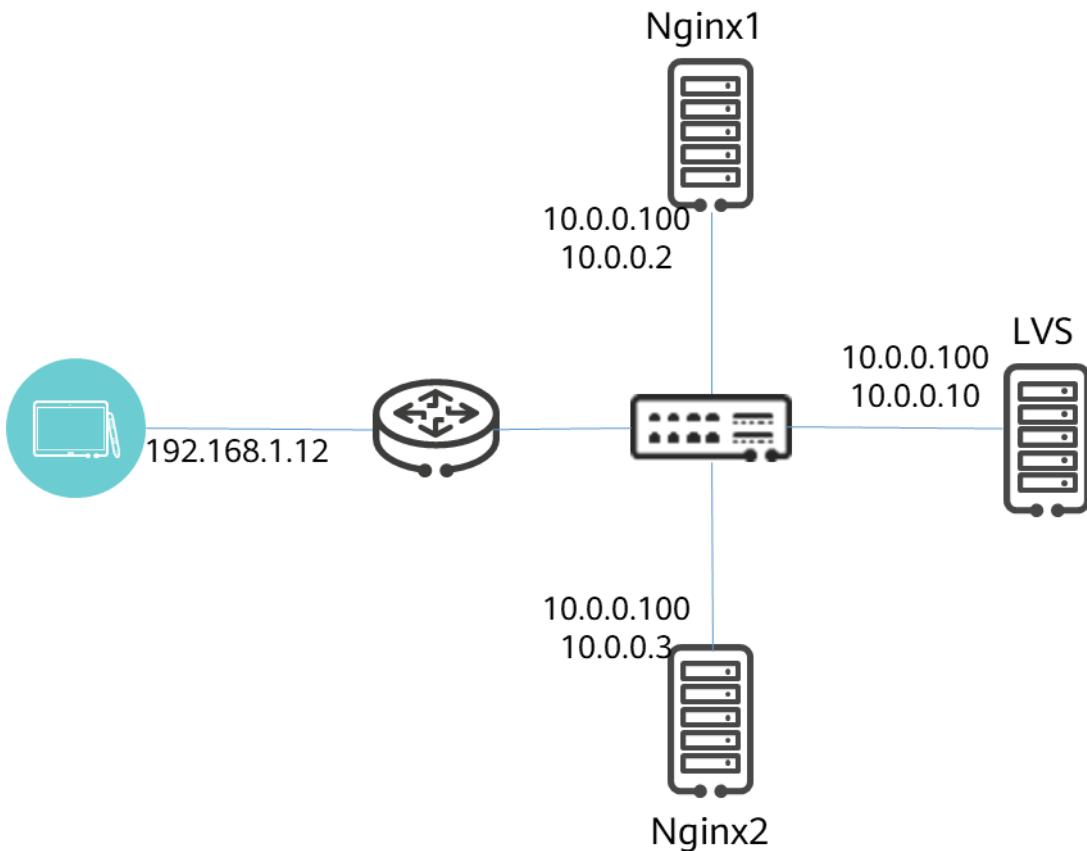
## 1.2 LVS-DR Configuration

### 1.2.1 Introduction

The requirements in this lab are basically the same as those in the preceding lab. The only difference is that the LVS working mode is changed from NAT to DR.

### 1.2.2 Networking

This lab requires four ECSs, one as the LVS, two as the NGINX servers, and the other as the client. The NGINX servers in the previous lab can be reused. The LVS requires only one NIC. Remove the NIC corresponding to 10.0.0.10 and change the IP address of the NIC where 192.168.1.10 is located to 10.0.0.10. The two NGINX servers and the LVS are in the same subnet, and the client is in another subnet. Configure dummy interfaces on the NGINX servers and the LVS to carry VIPs.



### 1.2.3 Procedure

#### Step 1 Configure Nginx1 and Nginx2.

The configuration method is the same as that in the previous lab.

Additionally, perform network configuration on the two NGINX nodes.

Run the following command to add VIP configurations:

```
nmcli connection add type dummy ifname dummy2 ipv4.method manual ipv4.addresses 10.0.0.100/32
```

Run the following commands to change the RIP gateway address to the router interface address:

```
nmcli connection modify ens224 ipv4.gateway 10.0.0.1  
nmcli connection down ens224  
nmcli connection up ens224
```

After the network configuration is complete, modify the ARP kernel parameters.

```
cat >> /etc/sysctl.conf << EOF  
net.ipv4.conf.all.arp_ignore = 1
```

```
net.ipv4.conf.all.arp_announce = 2  
net.ipv4.conf.dummy2.arp_ignore = 1  
net.ipv4.conf.dummy2.arp_announce = 2  
EOF
```

After the configuration is complete, run the **sysctl -p** command for the configuration to take effect.

```
[root@Nginx1 ~]# sysctl -p | tail -4  
net.ipv4.conf.all.arp_ignore = 1  
net.ipv4.conf.dummy2.arp_ignore = 1  
net.ipv4.conf.all.arp_announce = 2  
net.ipv4.conf.dummy2.arp_announce = 2
```

## Step 2 Preconfigure the LVS.

If the LVS in the previous lab is reused in this lab, remove the existing configurations.

Run the following command to delete the LVS configuration added in the preceding lab:

```
ipvsadm -D -t 192.168.1.10:80
```

After the deletion, run the following command to add VIP configurations:

```
nmcli connection add type dummy ifname dummy2 ipv4.method manual ipv4.addresses 10.0.0.100/32
```

Run the following commands to set the DIP gateway address to the router interface address:

```
nmcli connection modify ipv4.gateway 10.0.0.1  
nmcli connection down ens224  
nmcli connection up ens224
```

If the LVS is newly created, you only need to add the VIP configurations and change the gateway address.

## Step 3 Configure the LVS.

If the LVS is newly created, install ipvsadm based on Step 3 in 1.1.3.

Run the following commands to add required configurations:

```
ipvsadm -A -t 10.0.0.100:80 -s rr  
ipvsadm -a -t 10.0.0.100:80 -r 10.0.0.3  
ipvsadm -a -t 10.0.0.100:80 -r 10.0.0.2
```

Verify the configuration. Information in the following figure is displayed:

```
[root@Cluster1 ~]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  10.0.0.100:80 rr
    -> 10.0.0.2:80                 Route   1      0      0
    -> 10.0.0.3:80                 Route   1      0      0
```

- Question: What information in the preceding figure indicates that the current mode is DR?

Answer: **Route** in the **Forward** column.

#### Step 4 Check the LVS configuration.

Log in to the client ECS, use the LVS VIP to repeatedly access the web service. The following information is displayed:

```
[root@Client ~]# curl 10.0.0.100
hello,10.0.0.3
[root@Client ~]# curl 10.0.0.100
hello,10.0.0.2
[root@Client ~]# curl 10.0.0.100
hello,10.0.0.3
[root@Client ~]# curl 10.0.0.100
hello,10.0.0.2
[root@Client ~]#
```

It shows that the LVS forwards the requests to Nginx1 and Nginx2 in round robin mode and returns the page to the client.

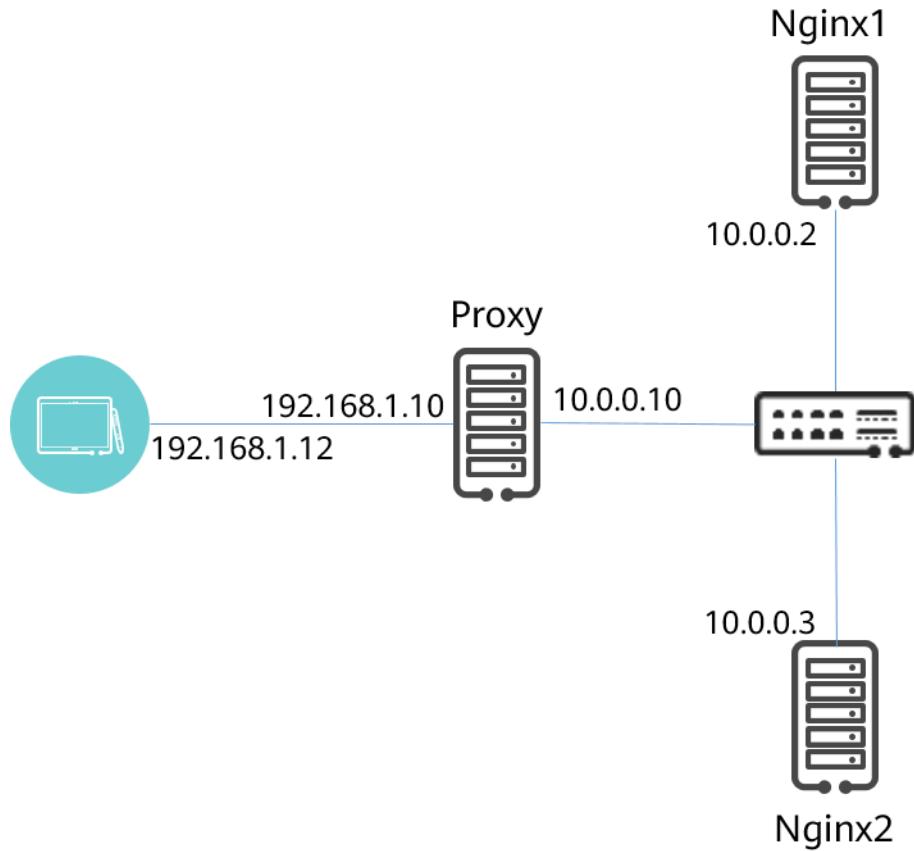
# 2 NGINX Reverse Proxy and Load Balancing Configuration

## 2.1 Introduction

This lab uses multiple NGINX algorithms to implement load balancing on backend servers and demonstrates the effect to enhance understanding of each algorithm.

## 2.2 Networking

In this lab, four ECSs are used. The two NGINX servers and the client in the LVS configuration lab can be reused. An NGINX proxy server needs be created. You can also reuse the LVS in the LVS configuration lab after clearing its configurations. The following figure shows the topology.



## 2.3 Procedure

Step 1 Check whether the resources are available.

On the client, ping the IP address in segment 10 of the proxy server and ensure that the network connection between them is normal, as shown in the following figure.

```
[root@Client ~]# ping 10.0.0.10
PING 10.0.0.10 (10.0.0.10) 56(84) bytes of data.
64 bytes from 10.0.0.10: icmp_seq=1 ttl=64 time=0.173 ms
64 bytes from 10.0.0.10: icmp_seq=2 ttl=64 time=0.123 ms
64 bytes from 10.0.0.10: icmp_seq=3 ttl=64 time=0.096 ms
```

Access the two NGINX servers on the proxy server to ensure that the NGINX servers are available, as shown in the following figure.

```
[root@Cluster1 conf.d]# curl 10.0.0.2
"hello,10.0.0.2"
[root@Cluster1 conf.d]# curl 10.0.0.3
"hello,10.0.0.3"
```

Step 2 Install and configure the NGINX proxy.

Install NGINX on the proxy server based on the knowledge in course 1, and then start the NGINX service, as shown in the following figure.

```
[root@Cluster1 conf.d]# systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: en
   Active: active (running) since Tue 2023-03-07 20:42:00
     Process: 9369 ExecStartPre=/usr/bin/rm -f /run/nginx.pid
     Process: 9372 ExecStartPre=/usr/sbin/nginx -t (code=exit, st
     Process: 9374 ExecStart=/usr/sbin/nginx (code=exited, st
```

Create a proxy and load balancing configuration file in the NGINX configuration file sub-directory of the proxy server and add the following configurations to the file:

```
upstream 10.0.0.10 {
    server 10.0.0.2:80;
    server 10.0.0.3:80;
}
server {
    listen 80;
    server_name localhost;
    location / {
        proxy_pass http://10.0.0.10;
    }
}
```

After the configuration, run the **nginx -s reload** command to reload the NGINX service.

### Step 3 Verify the proxy server configuration.

Access 10.0.0.1 from the client. If the configuration is correct, the following information is displayed:

```
[root@Client ~]# curl 10.0.0.10
hello,10.0.0.2
[root@Client ~]# curl 10.0.0.10
hello,10.0.0.3
[root@Client ~]# curl 10.0.0.10
hello,10.0.0.2
[root@Client ~]# curl 10.0.0.10
hello,10.0.0.3
```

### Step 4 Configure virtual hosts on Nginx1 and Nginx2.

To amplify the algorithm effect, add virtual hosts for the two NGINX servers.  
Configuration example:

```
server {
    listen 0.0.0.0:80;
    root /data/Nginx/;
    server_name localhost;
    index index80.html;
}
server {
    listen 0.0.0.0:81;
    root /data/Nginx/;
    server_name localhost;
    index index81.html;
}
server {
    listen 0.0.0.0:82;
    root /data/Nginx/;
    server_name localhost;
    index index82.html;
}
```

Create the **index80.html**, **index81.html**, and **index82.html** files in **/data/Nginx/**, and add the port to be used to the files.

```
[root@Client ~]# curl 10.0.0.2:81
"hello,10.0.0.2:81"
```

### Step 5 Add the virtual hosts to the backend host group.

Configuration example:

```
upstream 10.0.0.10 {
    server 10.0.0.2:80;
    server 10.0.0.2:81;
    server 10.0.0.2:82;
    server 10.0.0.3:80;
    server 10.0.0.3:81;
```

```
    server 10.0.0.3:82;
}
```

Reload the NGINX configuration file.

#### Step 6 Use the IP Hash algorithm for load balancing.

Modify to the configuration file of the proxy server as follows to change the algorithm to IP Hash:

```
upstream 10.0.0.10 {
    ip_hash;
    server 10.0.0.2:80;
    server 10.0.0.2:81;
    ....
}
....
```

Reload NGINX configurations and perform a test on the client. The result is as follows:

```
[root@Client ~]# curl 10.0.0.10
hello,10.0.0.3:81
[root@Client ~]# curl 10.0.0.10
hello,10.0.0.3:81
[root@Client ~]# curl 10.0.0.10
hello,10.0.0.3:81
```

#### Step 7 Use the Generic Hash algorithm for load balancing.

Create test files in the **root** directory on the two NGINX servers.

Run the following command on Nginx2:

```
[root@Nginx2 Nginx]# echo "10.0.0.3 Generic Hash practice" > test.txt
```

Run the following command on Nginx1:

```
[root@Nginx1 Nginx]# echo "10.0.0.2 Generic Hash practice" > test.txt
```

Reload the NGINX service.

To show the effect of the Generic Hash algorithm, use the default algorithm first.

```
upstream 10.0.0.10 {
    server 10.0.0.2:80;
    server 10.0.0.3:80;
    server 10.0.0.2:81;
    server 10.0.0.2:82;
```

Load NGINX configurations and access the **test.txt** file from the client. The result is as follows:

```
[root@Client ~]# curl 10.0.0.10/test.txt
10.0.0.2 Generic Hash practice
[root@Client ~]# curl 10.0.0.10/test.txt
10.0.0.2 Generic Hash practice
[root@Client ~]# curl 10.0.0.10/test.txt
10.0.0.3 Generic Hash practice
[root@Client ~]# curl 10.0.0.10/test.txt
10.0.0.3 Generic Hash practice
[root@Client ~]# curl 10.0.0.10/test.txt
10.0.0.2 Generic Hash practice
[root@Client ~]# curl 10.0.0.10/test.txt
10.0.0.3 Generic Hash practice
[root@Client ~]# curl 10.0.0.10/test.txt
10.0.0.2 Generic Hash practice
```

Modify the load balancing configuration file.

```
upstream 10.0.0.10 {
    hash $request_uri;
    server 10.0.0.2:80;
    server 10.0.0.3:80;
    server 10.0.0.2:81;
    server 10.0.0.2:82;
    server 10.0.0.3:81;
```

Reload NGINX configurations and access the **test.txt** file from the client. The result is as follows:

```
[root@Client ~]# curl 10.0.0.10/test.txt
10.0.0.3 Generic Hash practice
```

Step 8 Use the Random algorithm for load balancing.

Change the load balancing algorithm to Random as follows:

```
upstream 10.0.0.10 {  
    random two;  
    server 10.0.0.2:80;  
    server 10.0.0.3:80;  
    server 10.0.0.2:81;  
    server 10.0.0.2:82;  
    server 10.0.0.3:81;  
    server 10.0.0.3:82;
```

Reload NGINX configurations and perform a test. The result is as follows:

```
[root@Client ~]# curl 10.0.0.10  
hello,10.0.0.2:82  
[root@Client ~]# curl 10.0.0.10  
hello,10.0.0.2:81  
[root@Client ~]# curl 10.0.0.10  
hello,10.0.0.3:80  
[root@Client ~]# curl 10.0.0.10  
hello,10.0.0.2:81  
[root@Client ~]# curl 10.0.0.10  
hello,10.0.0.2:82  
[root@Client ~]# curl 10.0.0.10  
hello,10.0.0.3:81  
[root@Client ~]# curl 10.0.0.10  
hello,10.0.0.3:82  
[root@Client ~]# curl 10.0.0.10  
hello,10.0.0.2:81
```

- Question: After the algorithm is set to Random, will the test result be exactly the same as the preceding figure?

Answer: Not exactly. As the algorithm name indicates, the requests are scheduled to randomly selected servers.

#### Step 9 Configure algorithm options.

Modify the configuration file of the proxy server. Set all virtual hosts of Nginx2 to **backup** and assign a weight for the virtual hosts of Nginx1. Configuration example:

```
upstream 10.0.0.10 {  
    server 10.0.0.2:80 weight=2;  
    server 10.0.0.2:81;  
    server 10.0.0.2:82;  
    server 10.0.0.3:80 backup;  
    server 10.0.0.3:81 backup;  
    server 10.0.0.3:82 backup;
```

Reload NGINX configurations and perform a test on the client. The result is as follows:

```
[root@Client ~]# curl 10.0.0.10
hello,10.0.0.2:80
[root@Client ~]# curl 10.0.0.10
hello,10.0.0.2:80
[root@Client ~]# curl 10.0.0.10
hello,10.0.0.2:81
[root@Client ~]# curl 10.0.0.10
hello,10.0.0.2:82
[root@Client ~]# curl 10.0.0.10
hello,10.0.0.2:80
```

Manually stop the service on Nginx1, as shown in the following figure:

```
[root@Nginx1 ~]# systemctl stop nginx
[root@Nginx1 ~]# systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; vendor preset: disabled)
     Active: inactive (dead)
```

Perform the test on the client again. The result is as follows:

```
[root@Client ~]# curl 10.0.0.10
hello,10.0.0.3:80
[root@Client ~]# curl 10.0.0.10
hello,10.0.0.3:81
[root@Client ~]# curl 10.0.0.10
hello,10.0.0.3:82
[root@Client ~]# curl 10.0.0.10
hello,10.0.0.3:80
```

Modify the configuration file on the proxy server and change the status of some virtual hosts of Nginx2 to **down**. Configuration example:

```
upstream 10.0.0.10 {
    server 10.0.0.2:80 weight=2;
    server 10.0.0.2:81;
    server 10.0.0.2:82;
    server 10.0.0.3:80 down;
    server 10.0.0.3:81 backup;
    server 10.0.0.3:82 backup;
```

Reload NGINX configurations and perform a test on the client. The result is as follows:

```
[root@Client ~]# curl 10.0.0.10
hello,10.0.0.3:81
[root@Client ~]# curl 10.0.0.10
hello,10.0.0.3:82
[root@Client ~]# curl 10.0.0.10
hello,10.0.0.3:81
[root@Client ~]# curl 10.0.0.10
hello,10.0.0.3:82
```

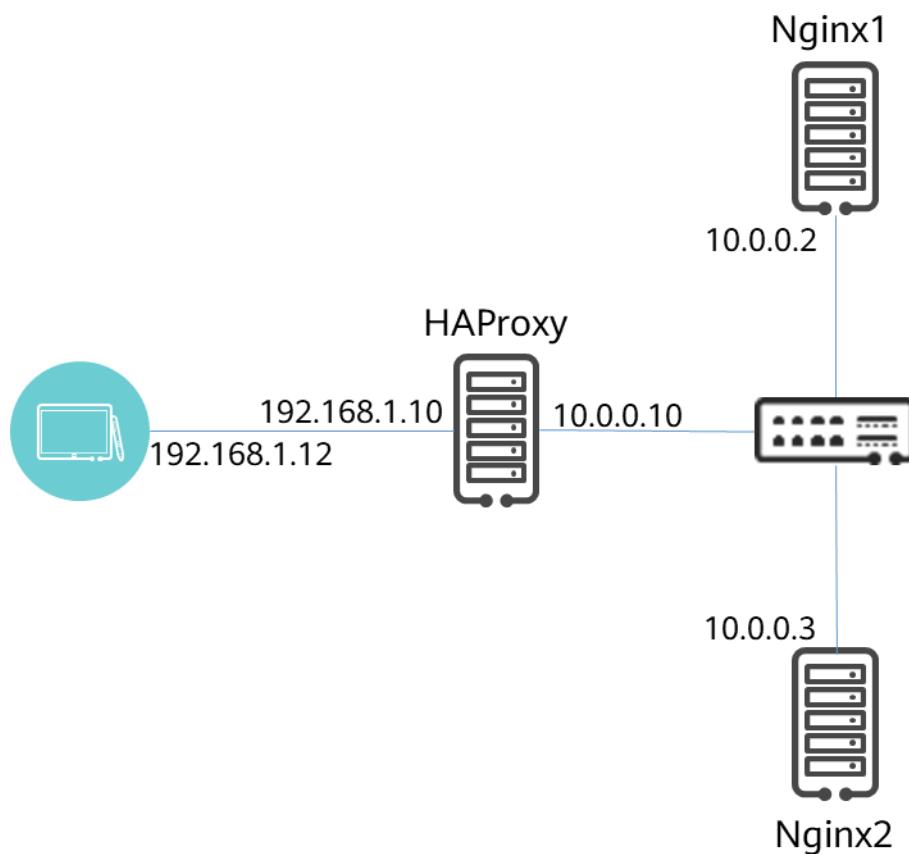
# 3 HAProxy Configuration

## 3.1 Introduction

This lab consists of multiple sub-labs to show various HAProxy functions.

## 3.2 Networking

This lab requires four ECSs, one as the HAProxy server, two as the NGINX servers, and the other as the client. The NGINX servers in the previous lab can be reused, and their configurations after the previous lab is complete can be retained. Each NGINX runs three virtual hosts. HAProxy uses two NICs to connect to the intranet and public network respectively. The following figure shows the topology.



## 3.3 Procedure

### 3.3.1 Implementing Basic Load Balancing with HAProxy

Step 1    Install HAProxy.

Run the following command on the HAProxy ECS to install HAProxy:

```
yum install -y haproxy
```

```
[root@Cluster1 ~]# yum install -y haproxy
Last metadata expiration check: 2:08:00 ago on
Dependencies resolved.

=====
           Package                     Architecture
=====
Installing:
haproxy                         x86_64
```

Step 2    Modify the HAProxy configuration file.

The HAProxy configuration file is **/etc/haproxy/haproxy.cfg**. You are advised to back up the file before modifying it.

This lab implements basic load balancing of HAProxy. You only need to modify the **backend** section as follows:

```
frontend main
  bind *:80
  default_backend      http_back

backend http_back
  balance    roundrobin
  server    node1 10.0.0.2:80 check
  server    node2 10.0.0.2:81 check
  server    node3 10.0.0.2:82 check
  server    node4 10.0.0.3:80 check
  server    node5 10.0.0.3:81 check
  server    node6 10.0.0.3:82 check
```

After the configuration, run the **systemctl restart haproxy** command to restart the HAProxy service. Then access HAProxy from the client. If the configuration is correct, the result is as follows:

```
[root@Client ~]# curl 192.168.1.10:80
hello,10.0.0.3:80
[root@Client ~]# curl 192.168.1.10:80
hello,10.0.0.3:81
[root@Client ~]# curl 192.168.1.10:80
hello,10.0.0.3:82
[root@Client ~]# curl 192.168.1.10:80
hello,10.0.0.3:80
[root@Client ~]# curl 192.168.1.10:80
hello,10.0.0.2:80
[root@Client ~]# curl 192.168.1.10:80
hello,10.0.0.2:81
[root@Client ~]# curl 192.168.1.10:80
hello,10.0.0.2:82
```

### 3.3.2 Configuring GUI-based HAProxy Monitoring

#### Step 1 Modify the HAProxy configuration file.

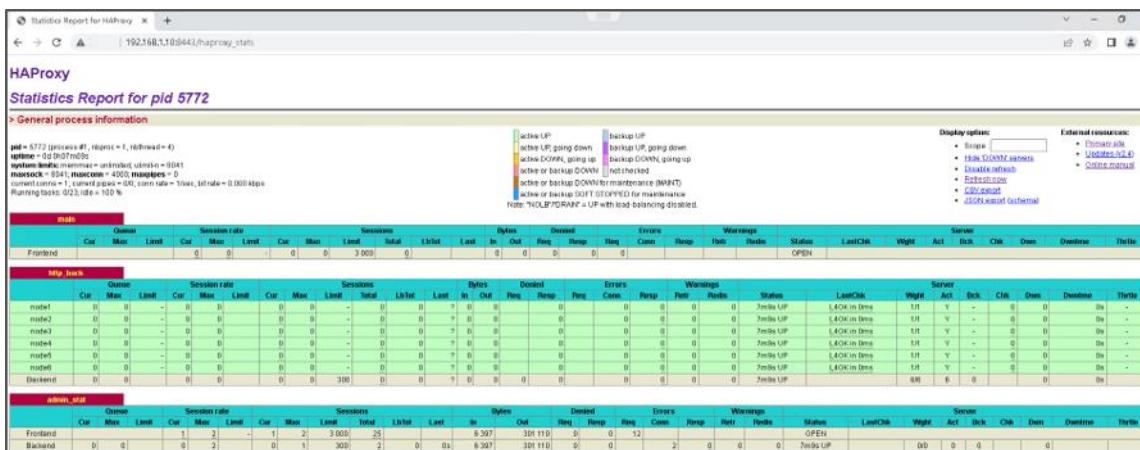
In the **listen** section in the HAProxy configuration file, configure GUI-based HAProxy monitoring.

```
listen admin_stat
    bind 0.0.0.0:8443
    mode http
    stats refresh 30s
    stats uri /haproxy_stats
    stats realm Haproxy\ Statistics
    stats auth openEuler:Huawei@123
    stats hide-version
```

**admin\_stat** indicates the user-defined name, **refresh 30s** indicates the refresh frequency, **uri /haproxy\_stats** indicates the URI to the monitoring page, and **openEuler:Huawei@123** indicates the authentication information for logging in to this page.

#### Step 2 View the monitoring page.

After the configuration is complete, restart the HAProxy service. Open a browser on the PC, enter **http://EIP:8443/haproxy\_stats** in the address box, and press **Enter**. In the displayed dialog box, enter the user name and password to log in to the monitoring page.



### 3.3.3 Configuring HAProxy Logs

#### Step 1 Modify the HAProxy configuration file.

HAProxy logs can be defined in the **global**, **default**, or **frontend** section. This lab uses **global** as an example. Add log configuration information to the HAProxy configuration file as follows:

```
global
  log      127.0.0.1 local3 info
  chroot   /var/lib/haproxy
  pidfile  /var/run/haproxy.pid
  user     haproxy
  group    haproxy
  daemon
  maxconn  4000
```

#### Step 2 Modify rsyslog configuration information.

Add the following contents to the end of **/etc/rsyslog.conf**:

```
local3.*                                     /var/log/haproxy.log
$ModLoad imudp
$UDPServerRun 514
```

#### Step 3 Perform a test.

After steps 1 and 2 are complete, restart the HAProxy and rsyslog services. After the restart, the system automatically creates a log file, as shown in the following figure.

```
[root@Cluster1 ~]# ls /var/log | grep haproxy
haproxy
haproxy.log
```

Enable the firewall on Nginx2 and check whether the corresponding logs are generated, as shown in the following figure.

```

Mar 9 10:09:00 localhost haproxy[5905]: Server http_back/node4 is DOWN, reason: Layer4 connection problem, info: "No route to host", check duration: 0ms. 5 active and 0 backup servers left. 0 sessions active, 0 requeued, 0 remaining in queue.
Mar 9 10:09:00 localhost haproxy[5905]: Server http_back/node5 is DOWN, reason: Layer4 connection problem, info: "No route to host", check duration: 0ms. 4 active and 0 backup servers left. 0 sessions active, 0 requeued, 0 remaining in queue.
Mar 9 10:09:00 localhost haproxy[5905]: Server http_back/node6 is DOWN, reason: Layer4 connection problem, info: "No route to host", check duration: 0ms. 3 active and 0 backup servers left. 0 sessions active, 0 requeued, 0 remaining in queue.
    
```

- Task: Log in to the monitoring page and check whether the status of the monitored hosts changes.

Expected result: The status of the monitored hosts changes to **DOWN**, as shown in the following figure.

	Queue			Session rate			Sessions			Bytes			Denied		Errors		Warnings		Status	LastChk		
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis		
node1	0	0	-	0	0	-	0	0	-	0	?	0	0	0	0	0	0	0	0	0	5m53s UP	L40K in 0ms
node2	0	0	-	0	0	-	0	0	-	0	0	?	0	0	0	0	0	0	0	0	5m53s UP	L40K in 0ms
node3	0	0	-	0	0	-	0	0	-	0	0	?	0	0	0	0	0	0	0	0	5m53s UP	L40K in 0ms
node4	0	0	-	0	0	-	0	0	-	0	0	?	0	0	0	0	0	0	0	0	3m14s DOWN	L4CON in 0ms
node5	0	0	-	0	0	-	0	0	-	0	0	?	0	0	0	0	0	0	0	0	3m14s DOWN	L4CON in 0ms
node6	0	0	-	0	0	-	0	0	-	0	0	?	0	0	0	0	0	0	0	0	3m14s DOWN	L4TOUT in 2001ms
Backend	0	0	-	0	0	-	0	0	300	0	0	?	0	0	0	0	0	0	0	0	5m53s UP	

#### Step 4 Restore Nginx2.

Disable the firewall service on Nginx2 to restore services.

### 3.3.4 Configuring HAProxy ACLs

In this lab, the URI ending with txt is sent to specified hosts. Configuration example:

```

frontend main
bind *:80
acl test url_reg -i \.txt$
use_backend test if test
default_backend          http_back
backend http_back
    balance    roundrobin
    server    node1 10.0.0.2:80 check
    server    node2 10.0.0.2:81 check
    server    node3 10.0.0.2:82 check
    server    node4 10.0.0.3:80 check
    server    node5 10.0.0.3:81 check
    server    node6 10.0.0.3:82 check
backend test
    balance    roundrobin
    server test1 10.0.0.2:80/test.txt check
    server test2 10.0.0.3:80/test.txt check
    
```

After the configuration, restart the HAProxy service and perform a test on the client. The result is as follows:

```
[root@Client ~]# curl 192.168.1.10/test.txt  
10.0.0.2 Generic Hash practice  
[root@Client ~]# curl 192.168.1.10/test.txt  
10.0.0.3 Generic Hash practice  
[root@Client ~]# curl 192.168.1.10/test.txt  
10.0.0.2 Generic Hash practice  
[root@Client ~]# curl 192.168.1.10/test.txt  
10.0.0.3 Generic Hash practice
```

- Question: What if **192.168.1.10/test.txt** is accessed with no ACL added?

Answer: HAProxy will forward the requests to the default backend host group in sequence. The result is as follows:

```
[root@Client ~]# curl 192.168.1.10/test.txt  
10.0.0.2 Generic Hash practice  
[root@Client ~]# curl 192.168.1.10/test.txt  
10.0.0.2 Generic Hash practice  
[root@Client ~]# curl 192.168.1.10/test.txt  
10.0.0.2 Generic Hash practice  
[root@Client ~]# curl 192.168.1.10/test.txt  
10.0.0.3 Generic Hash practice  
[root@Client ~]# curl 192.168.1.10/test.txt  
10.0.0.3 Generic Hash practice  
[root@Client ~]# curl 192.168.1.10/test.txt  
10.0.0.3 Generic Hash practice
```

# 4 Keepalived Configuration

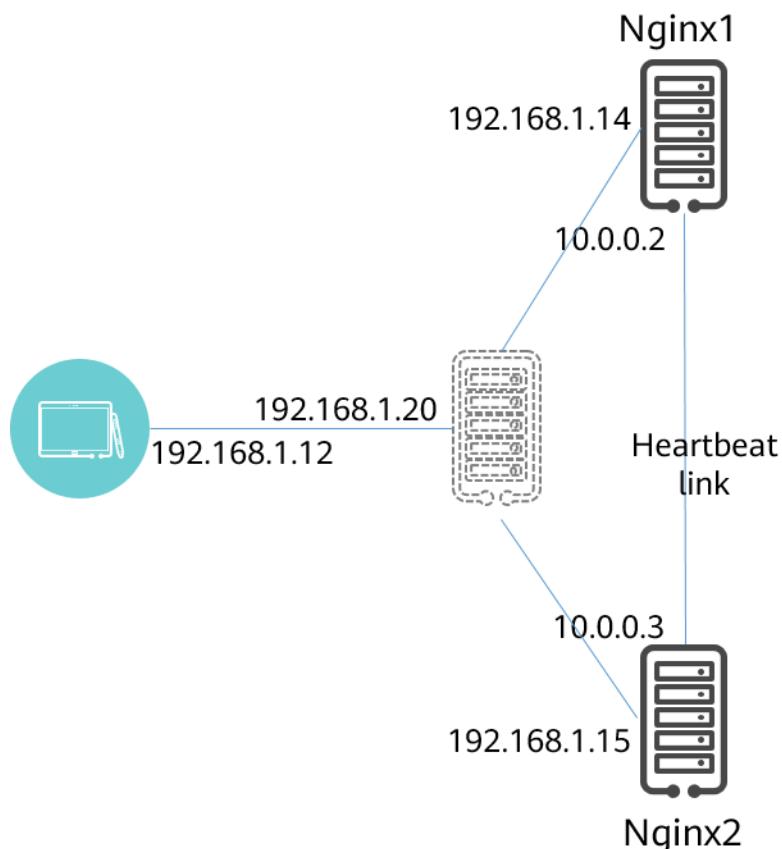
## 4.1 NGINX HA Cluster Configuration with Keepalived

### 4.1.1 Introduction

In this lab, Keepalived is used to implement an NGINX HA cluster. When the master server in the cluster is faulty, services are automatically switched to the backup server.

### 4.1.2 Networking

This lab requires three ECSs. Nginx1 and Nginx2 in the previous lab can be reused to run the NGINX service, and Keepalived is used to virtualize a host for client access. Nginx1 and Nginx2 detect each other's heartbeat through network segment 10 and provide services through floating IP address 192.168.1.20. If one host or the NGINX process on it is down, services are automatically switched to the other host. The following figure shows the topology.



Nginx1 and Nginx2 use two network interfaces. Configure them as planned, which will not be described in the following steps.

### 4.1.3 Procedure

#### Step 1 Install Keepalived.

Run the following command on Nginx1 and Nginx2 to install Keepalived:

```
yum install -y keepalived
```

```
[root@Nginx2 ~]# yum install -y keepalived
Last metadata expiration check: 0:17:32 ago on 2023-06-20 10:25:23 AM CST.
Dependencies resolved.

=====
| Package           | Architecture | Version |
|=====|
| Installing:      |             |          |
|   keepalived      | x86_64       | 2.2.4-2.oe2203 |
| Installing dependencies: |
|   lm_sensors      | x86_64       | 3.6.0-5.oe2203 |
|   mariadb-connector-c | x86_64       | 3.1.13-2.oe2203 |
```

#### Step 2 Check the network configurations of Nginx1 and Nginx2.

On Nginx1, ping Nginx2. If the network connection is normal, the following information is displayed:

```
[root@Nginx1 ~]# ping -I 10.0.0.2 10.0.0.3
PING 10.0.0.3 (10.0.0.3) from 10.0.0.2 : 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=0.328 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.252 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.324 ms
^C
--- 10.0.0.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2051ms
rtt min/avg/max/mdev = 0.252/0.301/0.328/0.034 ms
[root@Nginx1 ~]# ping -I 192.168.1.14 192.168.1.15
PING 192.168.1.15 (192.168.1.15) from 192.168.1.14 : 56(84) bytes of data.
64 bytes from 192.168.1.15: icmp_seq=1 ttl=64 time=0.352 ms
64 bytes from 192.168.1.15: icmp_seq=2 ttl=64 time=0.283 ms
^C
--- 192.168.1.15 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1038ms
rtt min/avg/max/mdev = 0.283/0.317/0.352/0.034 ms
```

#### Step 3 Configure Keepalived.

In this lab, Nginx1 is configured as the master node and Nginx2 is configured as the backup node. Modify the Keepalived configuration file (`/etc/keepalived/keepalived.conf`) on Nginx1 as follows:

```
! Configuration File for keepalived

global_defs {
    router_id Nginx1
}
```

```
vrrp_instance Nginx {
    state MASTER
    interface ens192
    virtual_router_id 51
    priority 225
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass Huawei@1
    }
    virtual_ipaddress {
        192.168.1.20/24
    }
}
```

Modify the Keepalived configuration file of Nginx2 as follows:

```
! Configuration File for keepalived

global_defs {
    router_id Nginx2
}

vrrp_instance Nginx {
    state BACKUP
    interface ens192
    virtual_router_id 51
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass Huawei@1
    }
    virtual_ipaddress {
        192.168.1.20/24
    }
}
```

After the configuration is complete, run the following command on Nginx1 and Nginx2 to restart the Keepalived service:

```
systemctl restart keepalived
```

Check whether a floating IP address is generated on Nginx1, as shown in the following figure.

```
[root@Nginx1 ~]# ip addr | grep 192
2: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq
    inet 192.168.1.14/24 brd 192.168.1.255 scope global
        inet 192.168.1.20/24 scope global secondary ens192
[root@Nginx1 ~]#
```

Step 4    Perform a test.

On the client, use the floating IP address to access the NGINX service. The result shows that Nginx1 responds to all the requests and returns the corresponding contents.

```
[root@Client ~]# curl 192.168.1.20
"hello,10.0.0.2"
[root@Client ~]# curl 192.168.1.20:81
"hello,10.0.0.2:81"
[root@Client ~]# curl 192.168.1.20:82
"hello,10.0.0.2:82"
```

Shut down Nginx1 and use the floating IP address to access NGINX again. The result is as follows:

```
[root@Client ~]# curl 192.168.1.20
"hello,10.0.0.3"
[root@Client ~]# curl 192.168.1.20:81
"hello,10.0.0.3:81"
[root@Client ~]# curl 192.168.1.20:82
"hello,10.0.0.3:82"
```

The floating IP address floats to Nginx2, as shown in the following figure.

```
[root@Nginx2 ~]# ip addr | grep 192
2: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP>
    inet 192.168.1.15/24 brd 192.168.1.255 scope global ens192
       inet 192.168.1.20/24 scope global secondary ens192
[root@Nginx2 ~]#
```

## Step 5 Configure a health check.

Currently, Keepalived can switch services only based on whether the master node breaks down. If only the NGINX service is faulty, Keepalived cannot switch services. Therefore, configure a health check in Keepalived to check whether NGINX is available.

Start Nginx1 that is shut down in step 4 and add health check configurations to the Keepalived configuration files of Nginx1 and Nginx2. The details are as follows:

```
global_defs {
    router_id Nginx1
}
vrrp_script nginx_check {
    script "/etc/keepalived/check.sh"
    interval 1
    weight -5
    fail 3
}
```

```
vrrp_instance Nginx {  
    .....  
    track_script {  
        nginx_check  
    }  
}
```

Create a **check.sh** script for checking NGINX service status in **/etc/keepalived** on the two NGINX servers. The script contents are as follows. Note: To prevent errors, do not directly copy and paste the following contents.

```
#!/bin/bash  
systemctl status nginx | grep "active (running)" > /dev/null  
if [ $? -ne 0 ]; then  
    systemctl restart nginx &> /dev/null  
    sleep 1  
    systemctl status nginx | grep "active (running)" > /dev/null  
    if [ $? -ne 0 ]; then  
        systemctl stop keepalived  
    else  
        exit  
    fi  
fi
```

## Step 6 Perform a test.

After the configuration is complete, run the **systemctl restart keepalived** command to restart the service. Modify the NGINX configuration file on Nginx1 so that the NGINX process cannot be started properly (for example, rename the **nginx.conf** file **nginx.conf.bk**). Then manually stop the NGINX service.

```
[root@Nginx1 ~]# mv /etc/nginx/nginx.conf /etc/nginx/nginx.conf.bk  
[root@Nginx1 ~]# systemctl stop nginx
```

Run the **systemctl start nginx** command and ensure that NGINX cannot be started, as shown in the following figure.

```
[root@Nginx1 ~]# systemctl start nginx  
Job for nginx.service failed because the control process exited with error code.  
See "systemctl status nginx.service" and "journalctl -xeu nginx.service" for details.
```

Perform the test on the client. It shows that Nginx2 provides services.

```
[root@Client ~]# curl 192.168.1.20  
"hello,10.0.0.3"  
[root@Client ~]# curl 192.168.1.20:81  
"hello,10.0.0.3:81"  
[root@Client ~]# curl 192.168.1.20:82  
"hello,10.0.0.3:82"
```

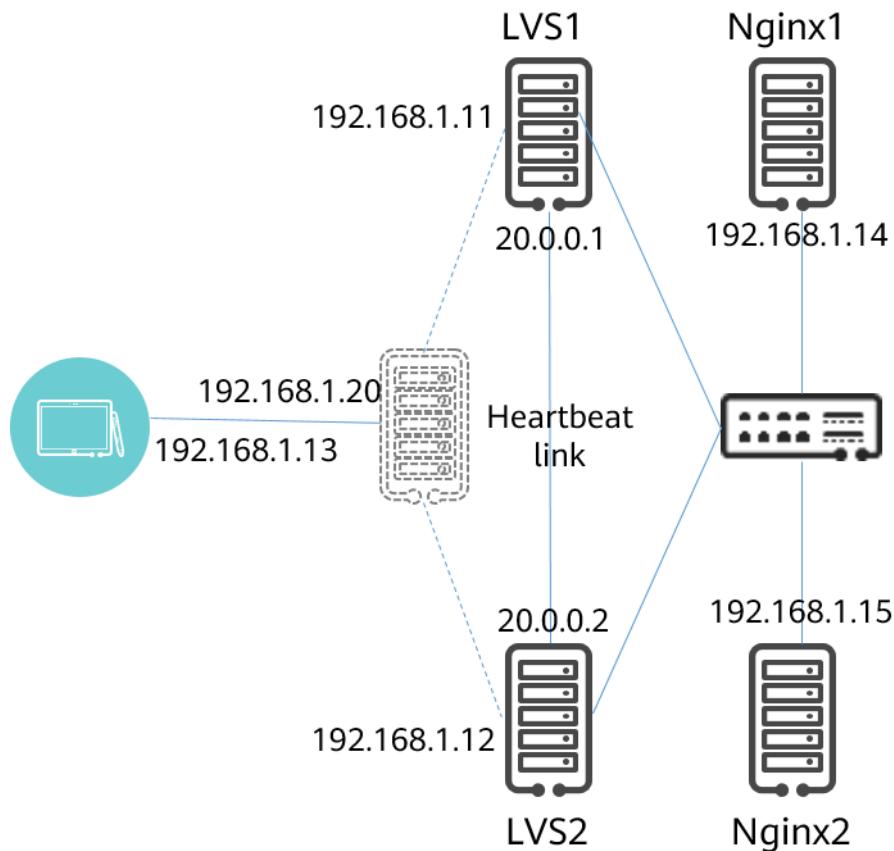
## 4.2 NGINX Cluster Configuration with Keepalived + LVS

### 4.2.1 Introduction

In this lab, an NGINX cluster is configured, where Keepalived provides HA configuration for the LVS and the LVS provides load balancing for backend servers Nginx1 and Nginx2.

### 4.2.2 Networking

This lab requires five ECSs. Nginx1 and Nginx2 in the previous lab can be reused, but you need to reconfigure IP addresses as planned in the topology below and uninstall Keepalived. IP address configuration procedure will not be described.



LVS1 and LVS2 form an HA cluster through Keepalived and provide load balancing for backend servers Nginx1 and Nginx2.

### 4.2.3 Procedure

**Step 1** Install Keepalived and ipvsadm on LVS1 and LVS2 ECSs.

Run the following command:

```
yum install -y keepalived ipvsadm
```

## Step 2 Modify the configuration file of LVS1.

Modify the Keepalived configuration file of LVS1 as follows:

```
! Configuration File for keepalived

global_defs {
    router_id Cluster1
}

vrrp_instance Nginx {
    state MASTER
    interface ens192
    mcast_src_ip 20.0.0.1
    virtual_router_id 51
    priority 255
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.1.20/24
    }
}

virtual_server 192.168.1.20 80 {
    delay_loop 6
    lb_algo rr
    lb_kind DR
    persistence_timeout 50
    protocol TCP

    real_server 192.168.1.14 80 {
        weight 1
        TCP_CHECK {
            connect_timeout 3
            retry 3
            delay_before_retry 3
        }
    }
    real_server 192.168.1.15 80 {
        weight 2
        TCP_CHECK {
            connect_timeout 3
            retry 3
            delay_before_retry 3
        }
    }
}
```

## Step 3 Modify the configuration file of LVS2.

Modify the Keepalived configuration file of LVS2 as follows:

```
! Configuration File for keepalived
```

```
global_defs {  
    router_id Cluster2  
}  
  
vrrp_instance Nginx {  
    state BACKUP  
    interface ens192  
    mcast_src_ip 20.0.0.2  
    virtual_router_id 51  
    priority 100  
    advert_int 1  
    authentication {  
        auth_type PASS  
        auth_pass 1111  
    }  
    virtual_ipaddress {  
        192.168.1.20/24  
    }  
}  
  
virtual_server 192.168.1.20 80 {  
    delay_loop 6  
    lb_algo rr  
    lb_kind DR  
    persistence_timeout 50  
    protocol TCP  
}  
  
real_server 192.168.1.14 80 {  
    weight 1  
    TCP_CHECK {  
        connect_timeout 3  
        retry 3  
        delay_before_retry 3  
    }  
}  
real_server 192.168.1.15 80 {  
    weight 2  
    TCP_CHECK {  
        connect_timeout 3  
        retry 3  
        delay_before_retry 3  
    }  
}
```

#### Step 4    Perform a test.

After the two LVS servers are configured, restart them and perform an access test on the client. The result is as follows:

```
[root@Client ~]# curl 192.168.1.20
"hello,10.0.0.4"
```

- Question: Why does Nginx2 respond to the requests every time?

Answer: Check the LVS configuration on LVS1. The details are as follows:

```
[root@Cluster1 keepalived]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
    -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  192.168.1.20:80 rr persistent 50
    -> 192.168.1.14:80             Route   1      0      0
    -> 192.168.1.15:80             Route   2      0      1
```

As shown in the preceding figure, LVS connections are configured as persistent connections by default, and the timeout interval is set to **50**. Therefore, each request is forwarded to the server allocated for the first time. You can comment out related configurations in the Keepalived configuration file to implement load balancing. See the following figure.

```
virtual_server 192.168.1.20 80 {
    delay_loop 6
    lb_algo rr
    lb_kind DR
    #persistence_timeout 50
    protocol TCP
    real_server 192.168.1.14 80 {
        weight 1
        TCP_CHECK {
            connect_timeout 3
            retry 3
    }}
```

Restart the Keepalived service and perform a test again. The test result is as follows:

```
[root@Client ~]# curl 192.168.1.20
"hello,10.0.0.4"
[root@Client ~]# curl 192.168.1.20
"hello,10.0.0.3"
[root@Client ~]# curl 192.168.1.20
"hello,10.0.0.4"
[root@Client ~]# curl 192.168.1.20
"hello,10.0.0.3"
```

# 5 Basic Redis Operations

## 5.1 Introduction

This lab shows how to install Redis in standalone mode and perform some simple Redis operations, such as creating, querying, and modifying keys.

## 5.2 Performing Basic Redis Operations

Step 1    Install Redis on an ECS.

Run the following command to install Redis:

```
yum install -y redis6-6.2.7-1.oe2203.x86_64
```

```
[root@Redis ~]# yum install -y redis6-6.2.7-1.oe2203.x86_64
Last metadata expiration check: 0:17:04 ago on Tue Aug 22 11:19:29 PM CST.
Dependencies resolved.
=====
 Package           Architecture      Version
=====
 Installing:
 redis6            x86_64          6.2.7-1.oe2203
 Transaction Summary
=====

```

After the installation is complete, run the `systemctl start redis` command to start the Redis service.

```
[root@Redis ~]# systemctl start redis
[root@Redis ~]# systemctl status redis
● redis.service - Redis persistent key-value database
   Loaded: loaded (/usr/lib/systemd/system/redis.serv
   Drop-In: /etc/systemd/system/redis.service.d
             └─limit.conf
     Active: active (running) since Tue Aug 22 23:3
               Main PID: 2275 (redis-server)
                 Status: "Ready to accept connections"
                Tasks: 5 (limit: 21624)
              Memory: 6.8M
             CGroup: /system.slice/redis.service
                       └─2275 "/usr/bin/redis-server 127.0.0.1:63
```

## Step 2 Log in to Redis and perform basic operations.

Run the **redis-cli** command to log in to Redis, as shown in the following figure.

```
[root@Redis ~]# redis-cli  
127.0.0.1:6379> [REDACTED]
```

Create data in key-value format.

```
set test1 openEuler1  
set test2 openEuler2  
set test3 openEuler3
```

```
127.0.0.1:6379> set test1 openEuler1  
OK  
127.0.0.1:6379> set test2 openEuler2  
OK  
127.0.0.1:6379> set test3 openEuler3  
OK
```

Query all keys in the current database.

```
keys *
```

```
127.0.0.1:6379> keys *  
1) "test3"  
2) "test2"  
3) "test1"
```

Run the **get** command to view the value of a key. For example:

```
127.0.0.1:6379> get test1  
"openEuler1"
```

Run the **expire** command to set the expiration time of a key. For example:

```
expire test1 2
```

```
127.0.0.1:6379> expire test1 2  
(integer) 1  
127.0.0.1:6379> get test1  
(nil)
```

Run the **move** command to migrate a key value to another database. For example:

```
127.0.0.1:6379> move test2 1  
(integer) 1  
127.0.0.1:6379> select 1  
OK  
127.0.0.1:6379[1]> keys *  
1) "test2"
```

- Question: Does the data still exist in Redis after the ECS is restarted in the current state?

Answer: Yes. By default, Redis automatically generates a **dump.rdb** file, which is a memory snapshot.

### Step 3 Set a Redis login password.

Modify the Redis configuration file **/etc/redis/redis.conf**. Uncomment **requirepass foobared** (line 903 in the current version) and replace **foobared** with a password, as shown in the following figure.

```
# The requirepass is not cor
# command, these will cause
#
requirepass Huawei@123
#
# New users are initialized
```

After the modification, restart the Redis service and check whether the password takes effect.

```
[root@Redis redis]# redis-cli
127.0.0.1:6379> keys *
(error) NOAUTH Authentication required.
127.0.0.1:6379> exit
[root@Redis redis]# redis-cli -a Huawei@123
Warning: Using a password with '-a' or '-u' option on the command line interface may not be safe
127.0.0.1:6379> keys *
1) "test3"
127.0.0.1:6379>
```

### Step 4 Configure Redis persistent storage.

Modify the Redis configuration file, set **dbfilename** to **snapshot.rdb**, and set **dir** to a directory that stores this file, as shown in the following figure.

```
# The filename where to dump the snapshot.
dbfilename snapshot.rdb

# Remove RDB files used by replication if replication is
# enabled. By default this option is disabled. It is useful
# where for regulations or other reasons it is not allowed to
# have disk by masters in order to free up space on the
# disk in order to load them for the slaves as soon as
# possible. Note that this option does not work if
# replication and RDB persistence disabled,
#
# An alternative (and sometimes better) way to handle
# this is to use diskless replication or to move
# the database in the case of replicas, diskless
# replication and RDB persistence disabled.
rdb-del-sync-files no

# The working directory.
#
# The DB will be written inside the specified directory
# above using the 'dbfilename' configuration option.
#
# The Append Only File will also be written inside
# the specified directory.
#
# Note that you must specify a directory, not a file name.
dir /var/lib/redis
```

Run the **save** command to set the RDB policy. In the following figure, **10** indicates that a snapshot is saved if one key changes within 10 seconds (line 386 in the configuration file).

```
# save 3600 1
# save 300 100
# save 60 10000
save 10 1
```

After the modification is complete, restart the Redis service, go to Redis, and manually create two keys. The system creates a snapshot file in the specified directory, as shown in the following figure.

```
[root@Redis ~]# ls /var/lib/redis/
dump.rdb snapshot.rdb
[root@Redis ~]#
```

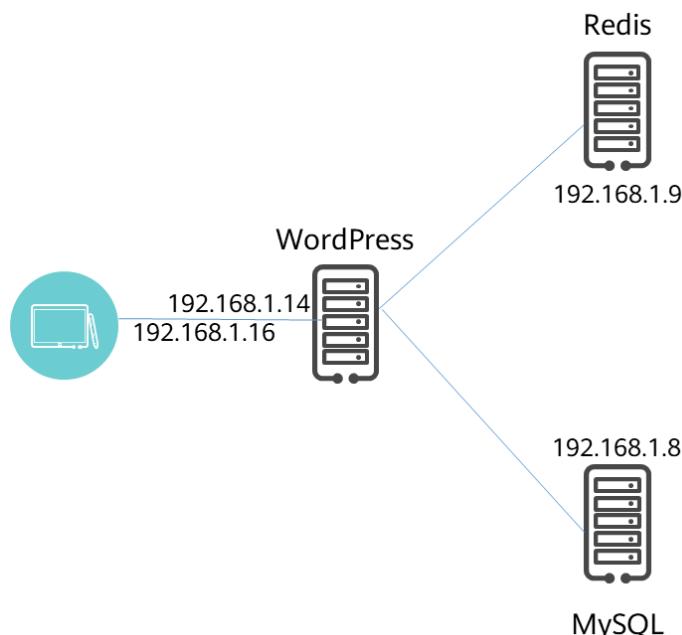
## 5.3 Configuring Redis as the Cache of WordPress

### 5.3.1 Introduction

In "LAMP Practices" in course 1, WordPress uses the HTTP service provided by Apache. In this lab, the LNMP architecture is used, that is, NGINX provides the HTTP service. In addition, Redis provides the cache for WordPress, together with WordPress plugins.

### 5.3.2 Networking

This lab requires four ECSs. Redis in the preceding lab can be reused, and Nginx1 in other labs can be reused as the WordPress server. A browser needs to be installed on the client. The following figure shows the topology and address planning.



### 5.3.3 Procedure

#### Step 1 Preconfigure Redis.

If Redis in the preceding lab is reused, delete all configurations.

```
[root@Redis ~]# redis-cli -a "Huawei@123"
Warning: Using a password with '-a' or '-u' option on the command line interface may not be safe.
127.0.0.1:6379> flushall
OK
127.0.0.1:6379> 
```

Delete snapshot files.

```
[root@Redis redis]# rm -rf /var/lib/redis/*
[root@Redis redis]# ls
```

Modify the configuration file (line 75) so that Redis can be accessed over the network.

```
74 # ~~~~~
75 bind 192.168.1.9
76
77 # Protected mode is a lave
```

After the configuration is complete, restart the server and check whether the configuration takes effect.

```
[root@Redis redis]# ss -lnp | grep redis
tcp    LISTEN  0      511                               192.168.1.9:6379
```

## Step 2 Interconnect NGINX with PHP.

On WordPress, run the **yum install -y php** command to install PHP and its dependencies.

After the installation is complete, add **listen = 9000** under **listen.allowed\_clients = 127.0.0.1** in the **/etc/php-fpm.d/www.conf** file, as shown in the following figure.

```
; must be separated by a comma. If there
; accepted from any ip address.
; Default Value: any
listen.allowed_clients = 127.0.0.1
listen = 9000
```

Run the **systemctl restart php-fpm** command to restart PHP and check whether port 9000 is enabled.

```
[root@Nginx1 conf.d]# ss -lnp | grep 9000
tcp    LISTEN  0      511                               *:9000
p-fpm",pid=1228,fd=9), ("php-fpm",pid=1227,fd=9), ("php-fpm",pid=1050,fd=7))
```

Create a **wordpress.conf** file in the **/etc/nginx/conf.d** directory of WordPress. The file content is as follows:

```
server {
    listen 0.0.0.0:80;
    root /data/WordPress;
    #server_name localhost;
    index index.php;
    location ~ \.php$ {
        fastcgi_pass 127.0.0.1:9000;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include         fastcgi_params;
    }
}
```

Create a directory **/data/WordPress**, create a test file **index.php** in this directory, and add the following contents to the file:

```
<?php
phpinfo();
?>
```

Restart the NGINX service and access the PHP page in the client browser to check whether the page can be displayed.



### Step 3 Interconnect PHP with MySQL.

This lab uses user **root** for testing. Check whether user **root** has the required permissions, as shown in the following figure.

```
mysql> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SELECT Host FROM user WHERE User='root';
+-----+
| Host |
+-----+
| %    |
+-----+
1 row in set (0.00 sec)
```

If user **root** is not allowed to access all other hosts or the WordPress host, complete the configuration by referring to the MySQL section.

After checking the permissions of user **root**, run the following commands on the WordPress host to install and connect to the MySQL server:

```
yum install -y mysql
mysql -h'192.168.1.8' -uroot -p'Huawei@123'
```

```
[root@Nginx1 data]# mysql -h'192.168.1.8' -uroot -p'Huawei@123'
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.28 Source distribution

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> exit
Bye
```

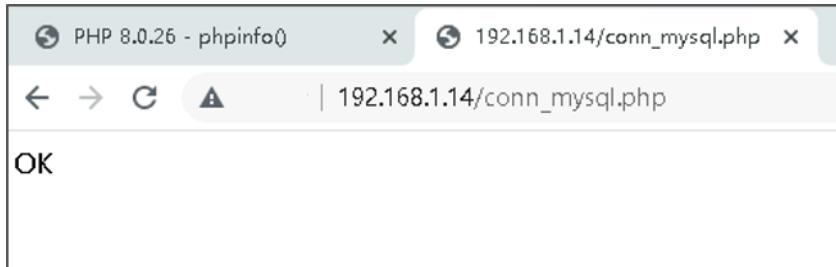
After the login is successful, run the following command on WordPress to install the PHP driver for MySQL:

```
yum install -y php-mysqlnd
```

After the installation is complete, create a **conn\_mysql.php** file in the **/data/WordPress** directory and enter the following content in the file:

```
<?php  
$con = mysqli_connect("192.168.1.8","root","Huawei@123");  
if ($con)  
    echo 'OK';  
else  
    echo 'NOT OK';  
$con->close();  
?>
```

After the configuration is complete, check whether **OK** is returned, as shown in the following figure.



#### Step 4 Prepare WordPress resources.

Download WordPress and create a database by following the instructions in "Preparing Resources" in course 1.

#### Step 5 Install WordPress.

Install and test WordPress by following the instructions in "Installing and Testing WordPress" in course 1.

#### Step 6 Install a Redis plugin.

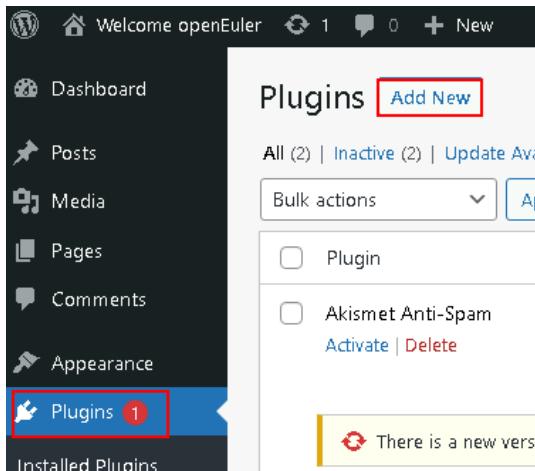
Add the following content to the end of the WordPress configuration file **wp-config**:

```
define("FS_METHOD", "direct");  
define("FS_CHMOD_DIR", 0777);  
define("FS_CHMOD_FILE", 0777);
```

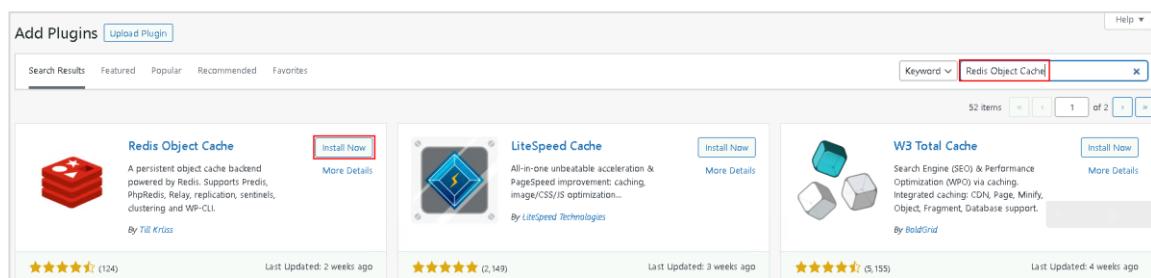
Run the following command to modify the permission on the WordPress directory:

```
chmod -R 777 WordPress
```

Log in to WordPress and click **Plugins > Add New**.



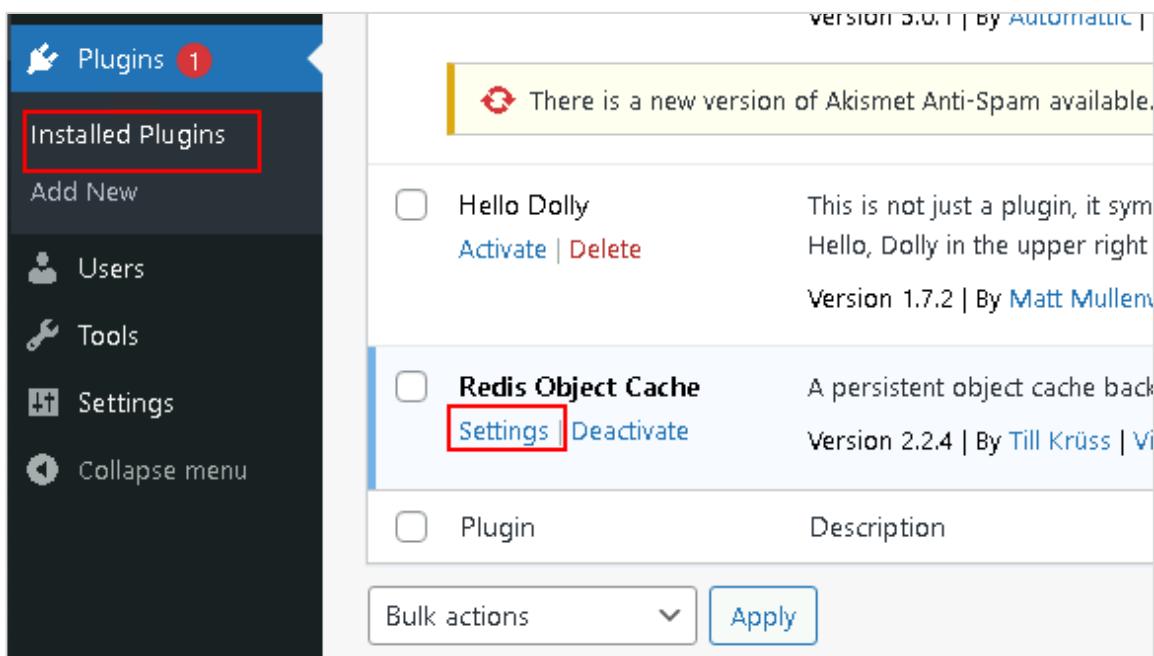
On the displayed page, enter **Redis Object Cache** in the search box and click **Install Now** in the search result, as shown in the following figure.



After the installation is complete, add the Redis configuration information to the WordPress configuration file **wp-config.php**. The details are as follows:

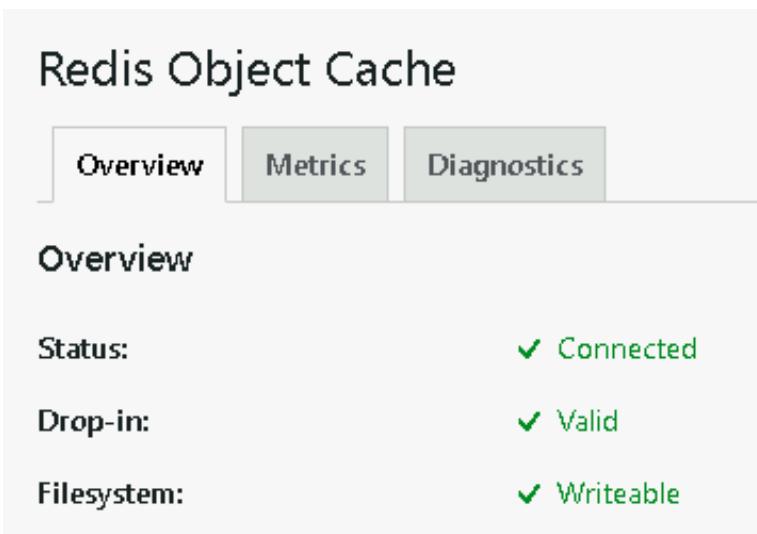
```
define( 'WP_REDIS_HOST', '192.168.1.9' );
define( 'WP_REDIS_PORT', 6379 );
define( 'WP_REDIS_PASSWORD', 'Huawei@123' );
define( 'WP_REDIS_TIMEOUT', 1 );
define( 'WP_REDIS_READ_TIMEOUT', 1 );
define( 'WP_REDIS_DATABASE', 0 );
```

Click **Activate** and then check the plugin status in **Installed Plugins**.



The screenshot shows the WordPress admin interface under the 'Plugins' section. A red box highlights the 'Installed Plugins' button in the sidebar. On the right, a message box indicates a new version of Akismet Anti-Spam is available. Below it, the 'Redis Object Cache' plugin is listed with its status as 'Active'. A red box highlights the 'Settings' link next to the plugin name. At the bottom, there are 'Bulk actions' and 'Apply' buttons.

If the following information is displayed, the plugin is running properly.



The screenshot shows the 'Redis Object Cache' overview page. It features three tabs: 'Overview' (selected), 'Metrics', and 'Diagnostics'. The 'Overview' section displays the following status information:

Status:	Connected
Drop-in:	Valid
Filesystem:	Writable

## Step 7    Query WordPress data stored in Redis.

Log in to the Redis host. In the WordPress configuration, the generated data is stored in database 0. Therefore, you can run the **keys \*** command to query the generated cache data.

```
[root@Redis ~]# redis-cli -h 192.168.1.9 -a Huawei@123
Warning: Using a password with '-a' or '-u' option on the command line interface may not be safe.
192.168.1.9:6379> keys *
1) "wp:post_tag_relationships:5"
2) "wp:site-transient:theme_roots"
3) "wp:posts:3"
4) "wp:post_format_relationships:5"
5) "wp:terms:get_terms-9fec3a134918b2d4e71f5916753f641e-0.97552500 1678848685"
6) "wp:posts:wp_query-8c6115e7f9d5fba29d4f161fd2728bae-0.87988800 16788484720.87889900 1678848472"
7) "wp:posts:wp_query-6b66f56649bde478b7401eb53d0d548c-0.11303000 1678848684"
8) "wp:options:uninstall_plugins"
9) "wp:options:alloptions"
10) "wp:terms:get_terms-f2e75d0a31510e7255fa16bc91a9e777-0.87889900 1678848472"
11) "wp:posts:wp_query-4cbfa75b1a602bcdd60106358247b7fc-0.96993300 16788486850.97552500 1678848685"
12) "wp:posts:5"
```

-----End-----

Huawei openEuler Certification Training

# HCIP-openEuler

## Storage Management

### Lab Guide

ISSUE: 1.0



HUAWEI TECHNOLOGIES CO., LTD

**Copyright © Huawei Technologies Co., Ltd. 2024. All rights reserved.**

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

### **Trademarks and Permissions**



HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

### **Notice**

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

## **Huawei Technologies Co., Ltd.**

Address:      Huawei Industrial Base Bantian, Longgang Shenzhen 518129  
                  People's Republic of China

Website:      <https://e.huawei.com>



## Huawei Certification System

Huawei Certification is an integral part of the company's Platform + Ecosystem strategy. It supports the development of ICT infrastructure that features Cloud-Pipe-Device synergy. Our certification is always evolving to reflect the latest trends in ICT development.

Huawei Certification consists of three categories: ICT Infrastructure Certification, Basic Software & Hardware Certification, and Cloud Platform & Services Certification, making it the most extensive technical certification program in the industry.

Huawei offers three levels of certification: Huawei Certified ICT Associate (HCIA), Huawei Certified ICT Professional (HCIP), and Huawei Certified ICT Expert (HCIE).

Our programs cover all ICT fields and follow the industry's trend of ICT convergence. With our leading talent development system and certification standards, we are committed to fostering new digital ICT talent and building a sound ICT talent ecosystem.

HCIP-openEuler is mainly for frontline engineers from Huawei and representative offices and readers who wish to learn openEuler O&M technologies. HCIP-openEuler certification covers common openEuler enterprise service management, openEuler HA cluster architecture, openEuler storage management, openEuler automated O&M, Linux shell scripts, openEuler system security hardening, and openEuler system monitoring.

Huawei certification helps you unlock opportunities to advance your career and take one more step towards the top of the industry.

## Huawei Career Certification

Cloud Platform & Services		Cloud Platform & Services Domain	Cloud Computing	Cloud Service	Big Data	GaussDB	IoT
Basic Software & Hardware		openEuler	openGauss	HarmonyOS	AI	Kunpeng	
ICT Infrastructure		IT Infrastructure Domain	Storage	Collaboration	Intelligent Vision	Digital Power	MDC
		Datacom Domain	Datacom		WLAN		Security
		Optical Domain		Transmission		Access	
		Wireless Domain	5G			LTE	
							Huawei Certified ICT Associate
							Huawei Certified ICT Professional
							Huawei Certified ICT Expert
							

# About This Document

---

## Overview

This document is an HCIP-openEuler certification training course and is intended for trainees who are going to take the HCIP-openEuler exam or readers who want to learn how to build enterprise services and master storage management on openEuler and other Linux distributions.

## Description

This lab guide consists of two labs, covering Network File System (NFS) and storage area network (SAN) on shared networks, as well as the configuration and implementation of GlusterFS distributed storage.

- Lab 1: shared storage configuration
  - This lab focuses on setting up and utilizing NFS and SAN storage, providing a foundation in basic shared storage usage.
- Lab 2: GlusterFS distributed storage management
  - This lab covers storage cluster setup, along with the creation and utilization of various volume types.

## Background Knowledge Required

This course is for Huawei certification. To better understand this course, you need to:

- Have basic Linux knowledge. You are advised to complete HCIA-openEuler learning and pass the HCIA-openEuler certification exam.

## Lab Environment Overview

This experiment is based on Huawei Cloud Elastic Cloud Server (ECS). Some lab resources are utilized across different labs. You can choose to create new ECSs or reuse existing ones as needed.

For details about the lab environment parameters of each lab, see the corresponding lab networking guide and lab resource list.

You can refer to the lab guide of the HCIA-Cloud Service certification course for the basic usage of Huawei Cloud.



# Lab Environment Preparation

## Checking Devices

Prepare a Huawei Cloud account with a sufficient budget.

# Contents

---

<b>About This Document .....</b>	<b>3</b>
Overview .....	3
Description .....	3
Background Knowledge Required .....	3
Lab Environment Overview.....	3
Lab Environment Preparation .....	4
<b>1 Shared Storage Management.....</b>	<b>1</b>
1.1 NFS Storage Management.....	1
1.1.1 Introduction .....	1
1.1.2 Networking.....	1
1.1.3 Procedure.....	2
1.2 SAN Storage .....	9
1.2.1 Introduction .....	9
1.2.2 Networking.....	9
1.2.3 Procedure .....	10
<b>2 GlusterFS Distributed Storage.....</b>	<b>33</b>
2.1 Introduction .....	33
2.2 Networking .....	33
2.3 Procedure .....	35

# 1

# Shared Storage Management

---

## 1.1 NFS Storage Management

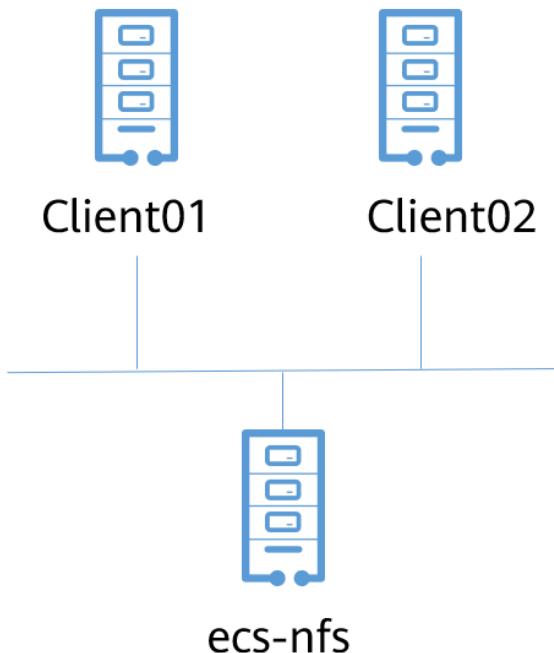
### 1.1.1 Introduction

In this lab, you will set up an NFS server on a Huawei Cloud ECS and mount the shared storage to another two ECS for access. Lab operations include:

- NFS server setup
- NFS client setup
- Mounting of volumes and use of NFS shared storage on a client
- NFS access control
- NFS automatic mounting configuration

### 1.1.2 Networking

This lab involves three ECSs: **Client01**, **Client02**, and **ecs-nfs**. The following figure shows the networking.



ECS Name	IP Address	Specifications
ecs-nfs	192.168.1.10	OS: openEuler 22.03 LTS
Client01	192.168.1.21	Password: Set a password as needed. Flavor: 2 vCPUs   4 GB Drive: 40 GB
Client02	192.168.1.22	Each ECS is bound to an EIP.

### 1.1.3 Procedure

#### Step 1 Create ECSs.

Create three ECSs based on the lab networking.

#### Step 2 Install the NFS server.

Log in to **ecs-nfs** and check whether nfs-utils and rpcbind have been installed.

```
rpm -aq | grep nfs
rpm -aq | grep rpcbind
```

```
[root@ecs-nfs ~]# rpm -aq |grep nfs
nfs-utils-help-2.5.4-4.oe2203.x86_64
nfs-utils-2.5.4-4.oe2203.x86_64
[root@ecs-nfs ~]# rpm -aq |grep rpcbind
rpcbind-1.2.6-3.oe2203.x86_64
[root@ecs-nfs ~]#
```

The private OS image used in this lab contains the nfs-utils and rpcbind software packages, and the services have been installed. You only need to start the services.

If the nfs or rpcbind process does not exist in the environment, manually install nfs-utils. rpcbind will be automatically installed with nfs-utils.

```
yum install nfs-utils
```

After the installation is complete, check the NFS service status.

```
systemctl status nfs
```

```
[root@ecs-nfs ~]# systemctl status nfs
● nfs-server.service - NFS server and services
  Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; disabled; vendor preset: disabled)
    Active: inactive (dead)
[root@ecs-nfs ~]#
```

Start the nfs service and enable it to automatically start upon system startup.

```
systemctl enable nfs
```

```
systemctl start nfs
```

```
[root@ecs-nfs ~]# systemctl enable nfs
Created symlink /etc/systemd/system/multi-user.target.wants/nfs-server.service → /usr/lib/systemd/system/nfs-server.service.
[root@ecs-nfs ~]# systemctl start nfs
```

Check the NFS service status again.

```
[root@ecs-nfs ~]# systemctl status nfs
● nfs-server.service - NFS server and services
  Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; enabled; vendor preset: disabled)
    Active: active (exited) since Sun 2023-04-23 15:36:03 CST; 33s ago
      Process: 1861 ExecStartPre=/usr/sbin/exportfs -r (code=exited, status=0/SUCCESS)
      Process: 1863 ExecStart=/usr/sbin/rpc.nfsd (code=exited, status=0/SUCCESS)
        Main PID: 1863 (code=exited, status=0/SUCCESS)

Apr 23 15:36:03 ecs-nfs systemd[1]: Starting NFS server and services...
Apr 23 15:36:03 ecs-nfs systemd[1]: Finished NFS server and services.
```

### Step 3 Configure an NFS shared directory.

Create a shared directory **/data** and check whether the creation is successful. The shared directory will be mounted and accessed by clients on other ECSS.

```
mkdir /data
ls /
```

```
[root@ecs-nfs ~]# mkdir /data
[root@ecs-nfs ~]# ls /
afs  boot          CloudrResetPwdAgent  dev   home  lib64  media  opt   root  sbin  sys  usr
bin  CloudResetPwdUpdateAgent  data    etc   lib    lost+found  mnt   proc  run   srv   tmp  var
[root@ecs-nfs ~]#
```

Edit the **/etc/exports** configuration file to allow access from any clients.

```
/data *(rw,sync,no_root_squash)
```

```
[root@ecs-nfs ~]# vi /etc/exports
[root@ecs-nfs ~]#
[root@ecs-nfs ~]#
[root@ecs-nfs ~]# cat /etc/exports
/data *(rw,sync,no_root_squash)
[root@ecs-nfs ~]#
```

After the configuration is complete, restart the nfs service.

```
systemctl restart nfs
```

```
[root@ecs-nfs ~]# systemctl restart nfs
[root@ecs-nfs ~]#
```

Check whether the shared directory exists.

```
showmount -e
```

```
[root@ecs-nfs ~]# showmount -e  
Export list for ecs-nfs:  
/data *
```

#### Step 4 Configure the client.

Log in to **ECS Client01** and install the nfs-utils and rpcbind services by referring to Step 2.

```
[root@Client01 ~]# systemctl status nfs  
● nfs-server.service - NFS server and services  
  Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; enabled; vendor preset: disabled)  
  Active: active (exited) since Sun 2023-04-23 15:44:24 CST; 3s ago  
    Process: 1794 ExecStartPre=/usr/sbin/exportfs -r (code=exited, status=0/SUCCESS)  
    Process: 1795 ExecStart=/usr/sbin/rpc.nfsd (code=exited, status=0/SUCCESS)  
   Main PID: 1795 (code=exited, status=0/SUCCESS)  
  
Apr 23 15:44:23 Client01 systemd[1]: Starting NFS server and services...  
Apr 23 15:44:24 Client01 systemd[1]: Finished NFS server and services.  
[root@Client01 ~]#
```

In this lab, **Client01** and **ecs-nfs** are on the same network and are mutually reachable.

```
[root@Client01 ~]# ping 192.168.1.10  
PING 192.168.1.10 (192.168.1.10) 56(84) bytes of data.  
64 bytes from 192.168.1.10: icmp_seq=1 ttl=64 time=1.09 ms  
64 bytes from 192.168.1.10: icmp_seq=2 ttl=64 time=0.210 ms
```

Proceed to Step 5. If the ECSs cannot reach each other, establish connectivity between them before performing subsequent operations.

#### Step 5 Mount the shared directory to the client.

On **Client01**, create mount point **/mnt/data**.

```
mkdir /mnt/data
```

```
[root@Client01 ~]# mkdir /mnt/data  
[root@Client01 ~]# ls /mnt  
data
```

Check whether the shared directory is available on **ecs-nfs**.

```
showmount -e 192.168.1.10
```

```
[root@Client01 ~]# showmount -e 192.168.1.10  
Export list for 192.168.1.10:  
/data *
```

If the shared directory exists, mount it to the **/mnt/data** directory on the client.

```
mount -t nfs 192.168.1.10:/data /mnt/data
```

```
[root@Client01 ~]# mount -t nfs 192.168.1.10:/data /mnt/data  
[root@Client01 ~]#
```

Successful mounting produces no output. Run the **df -h** command to check the status of storage on **Client01**.

```
[root@Client01 ~]# df -h  
Filesystem           Size   Used  Avail Use% Mounted on  
devtmpfs             1.7G    0     1.7G  0% /dev  
tmpfs               1.7G    0     1.7G  0% /dev/shm  
tmpfs               683M   8.6M  674M  2% /run  
tmpfs               4.0M    0     4.0M  0% /sys/fs/cgroup  
/dev/mapper/openeuler-root  35G   2.5G  30G  8% /  
tmpfs               1.7G   32K   1.7G  1% /tmp  
/dev/vda1            974M   87M  820M 10% /boot  
192.168.1.10:/data      35G   2.5G  30G  8% /mnt/data  
[root@Client01 ~]#
```

Step 6     Uses the NFS shared directory on the client.

On **Client01**, enter the mount point and create **file01** in the shared directory.

```
[root@Client01 ~]# cd /mnt/data  
[root@Client01 data]# touch file01  
[root@Client01 data]# ls  
file01  
[root@Client01 data]#
```

Write "*This is from Client01!*" in **file01**.

```
[root@Client01 data]# cat file01  
This is from Client01!
```

On **ecs-nfs**, go to the **/data** directory and check whether the preceding operation is successful.

```
[root@ecs-nfs ~]# cd /data  
[root@ecs-nfs data]# ls  
file01  
[root@ecs-nfs data]# cat file01  
This is from Client01!  
[root@ecs-nfs data]#
```

Log in to **Client02** and repeat steps 4 and 5 to mount the NFS shared directory.

```
[root@Client02 ~]# df -h
Filesystem           Size  Used  Avail Use% Mounted on
devtmpfs             1.7G   0    1.7G  0% /dev
tmpfs                1.7G   0    1.7G  0% /dev/shm
tmpfs                683M  8.6M  674M  2% /run
tmpfs                4.0M   0    4.0M  0% /sys/fs/cgroup
/dev/mapper/openeuler-root  35G  2.5G  30G  8% /
tmpfs                1.7G  32K  1.7G  1% /tmp
/dev/vda1              974M  87M  820M 10% /boot
192.168.1.10:/data      35G  2.5G  30G  8% /mnt/data
[root@Client02 ~]#
```

Step 7 Access files in the share directory from a client.

On Client02, go to the mount directory and view files in the directory.

```
[root@Client02 ~]# cd /mnt/data
[root@Client02 data]# ls
file01
[root@Client02 data]# cat file01
This is from Client01!
[root@Client02 data]#
```

Modify **file01** as follows:

```
[root@Client02 data]# vi file01
[root@Client02 data]#
[root@Client02 data]#
[root@Client02 data]# cat file01
This is from Client02!

[root@Client02 data]#
```

On Client01, check the file content.

```
[root@Client01 data]# cat file01
This is from Client02!
```

You can see that the file has been modified.

Step 8 Configure access control on the NFS shared directory.

On **ecs-nfs**, modify the **/etc/exports** file to refine client access permissions and restart the NFS service.

```
/data 192.168.1.21/32(rw,sync,no_root_squash)
/data 192.168.1.22/32(ro)
```



```
[root@ecs-nfs ~]# cat /etc/exports  
/data 192.168.1.21/32(rw,sync,no_root_squash)  
/data 192.168.1.22/32(ro)
```

```
[root@ecs-nfs data]# systemctl restart nfs  
[root@ecs-nfs data]#
```

On **Client01**, modify **file01** in the shared directory as follows.

```
[root@Client01 data]# cat file01  
Change from Client01!
```

On **Client02**, modify **file01** in the shared directory as follows.

Change from Client02!

Z Z Z Z Z Z Z Z

E45: 'readonly' option is set (add ! to override)

You can see that **Client02** has only the read-only permission on the file and cannot modify it. Try creating files in the shared directory. You can see that **Client02** does not have the permission to create files, while **Client01** does.

```
[root@Client02 data]# ls  
file01  
[root@Client02 data]# mkdir file02  
mkdir: cannot create directory 'file02': Read-only  
file system  
[root@Client02 data]#
```

```
[root@Client01 data]# mkdir file01-2  
[root@Client01 data]# ls  
file01  file01-2  
[root@Client01 data]#
```

## Step 9 Set automatic mounting upon system startup.

Restart **Client01** and check its storage information.

```
[root@Client01 ~]# df -h
Filesystem           Size  Used  Avail Use% Mounted on
devtmpfs             1.7G   0    1.7G  0% /dev
tmpfs               1.7G   0    1.7G  0% /dev/shm
tmpfs               683M  8.6M  674M  2% /run
tmpfs               4.0M   0    4.0M  0% /sys/fs/cgroup
/dev/mapper/openeuler-root  35G  2.5G  30G  8% /
tmpfs               1.7G  32K  1.7G  1% /tmp
/dev/vda1            974M  87M  820M 10% /boot
```

After the restart, the shared directory is unmounted. Configure automatic mounting.

On **Client01**, add the following content to the **/etc/fstab** file, and save the file.

```
192.168.1.10:/data /mnt/data nfs defaults 0 0
```

```
# 
# /etc/fstab
# Created by anaconda on Mon Apr  3 08:21:21 2023
#
# Accessible filesystems, by reference, are maintained under '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
/dev/mapper/openeuler-root /          ext4      defaults        1 1
UUID=aeae26a7d-9b83-4a0d-b219-ae8a0191cd72 /boot      ext4      defaults        1 2
/dev/mapper/openeuler-swap none       swap      defaults        0 0
192.168.1.10:/data /mnt/data nfs defaults 0 0
~
```

Restart **Client01** again and check its storage information. You can see that the shared directory is automatically mounted.

```
[root@Client01 ~]# df -h
Filesystem           Size  Used  Avail Use% Mounted on
devtmpfs             1.7G   0    1.7G  0% /dev
tmpfs               1.7G   0    1.7G  0% /dev/shm
tmpfs               683M  8.6M  674M  2% /run
tmpfs               4.0M   0    4.0M  0% /sys/fs/cgroup
/dev/mapper/openeuler-root  35G  2.5G  30G  8% /
tmpfs               1.7G  32K  1.7G  1% /tmp
/dev/vda1            974M  87M  820M 10% /boot
192.168.1.10:/data  35G  2.5G  30G  8% /mnt/data
```

## 1.2 SAN Storage

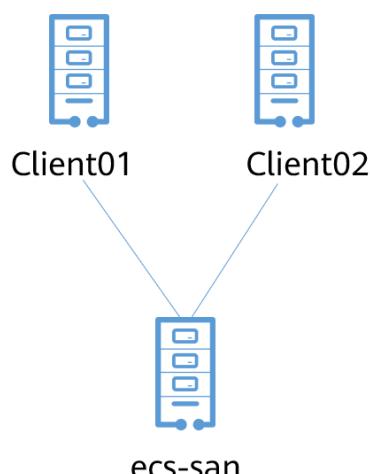
### 1.2.1 Introduction

In this lab, you will reuse **Client01** and **Client02** from the NFS storage management lab as clients to run initiator processes. You will also create a separate ECS to simulate setup and usage of IP-SAN storage. Lab operations include:

- IP-SAN storage server setup
- Client setup and storage mounting
- Shared storage access and read/write

### 1.2.2 Networking

This lab involves three ECSs: **Client01** and **Client02** (from the NFS storage management lab) as clients, and a separate ECS as the simulated IP-SAN storage. The following figure shows the networking.



ECS Name	IP Address	Remarks	Specifications
ecs-san	192.168.1.3	SAN server	OS: openEuler 22.03 LTS Password: Set a password as needed. Flavor: 2 vCPUs   4 GB System drive: 40 GB Data drive: 10 GB Each ECS is bound to an EIP.
Client01	192.168.1.21	Reused	
Client02	192.168.1.22	Reused	

## 1.2.3 Procedure

Step 1 Log in to the Huawei Cloud official website and create ECS **ecs-san**.

Step 2 Install the iSCSI software package on the SAN server.

Log in to **ecs-san** and install the target software package.

```
yum -y install scsi-target-utils
```

```
[root@ecs-san ~]# yum -y install scsi-target-utils
OS
everything
EPOL
debuginfo
source
update
Dependencies resolved.
=====
| Package           | Architecture | Version      | Repository | Size
=====
| Installing:      |             |             |            |
|   scsi-target-utils | x86_64      | 1.0.79-4.oe2203 | everything | 193 k
| Installing dependencies: |
|   lsof             | x86_64      | 4.94.0-1.oe2203 | 05          | 93 k
|   perl-Config-General | noarch     | 2.63-1.oe2203  | everything | 39 k
|   rdma-core        | x86_64      | 35.1-2.oe2203  | 05          | 795 k
|
Transaction Summary
=====
Install 4 Packages
=====
Total download size: 1.1 M
Installed size: 3.9 M
Downloading Packages:
(1/4): lsof-4.94.0-1.oe2203.x86_64.rpm                                              155 kB/s |  93 kB  00:00
(2/4): perl-Config-General-2.63-1.oe2203.noarch.rpm                                     27 kB/s |  39 kB  00:01
(3/4): scsi-target-utils-1.0.79-4.oe2203.x86_64.rpm                                    110 kB/s | 193 kB  00:01
[MIRROR] rdma-core-35.1-2.oe2203.x86_64.rpm: Curl error (28): Timeout was reached for http://repo.openeuler.org/openEuler-22.03-LTS/x86_64/Packages/rdma-core-35.1-2.oe2203.x86_64.rpm [Connection timeout after 30000 ms]
(4/4): rdma-core-35.1-2.oe2203.x86_64.rpm                                              23 kB/s | 795 kB  00:34
=====
33 kB/s | 1.1 MB  00:34
Total
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing : 1/1
  Installing : perl-Config-General-2.63-1.oe2203.noarch 1/1
  Installing : rdma-core-35.1-2.oe2203.x86_64 2/4
  Running transaction scriptlet: 2/4
  Installing : lsof-4.94.0-1.oe2203.x86_64 3/4
  Installing : scsi-target-utils-1.0.79-4.oe2203.x86_64 4/4
  Running scriptlet: scsi-target-utils-1.0.79-4.oe2203.x86_64 4/4
  Verifying  : lsof-4.94.0-1.oe2203.x86_64 4/4
  Verifying  : rdma-core-35.1-2.oe2203.x86_64 1/4
  Verifying  : perl-Config-General-2.63-1.oe2203.noarch 2/4
  Verifying  : scsi-target-utils-1.0.79-4.oe2203.x86_64 3/4
  Complete! 4/4
=====
1/1
1/1
2/4
3/4
3/4
4/4
4/4
4/4
1/4
2/4
3/4
4/4
```

Check the generated files.

```
rpm -ql scsi-target-utils
```

```
[root@ecs-san ~]# rpm -ql scsi-target-utils
/etc/ima/digest_lists.tlv/0-metadata_list-compact_tlv-scsi-target-utils-1.0.79-4.oe2203.x86_64
/etc/ima/digest_lists/0-metadata_list-compact-scsi-target-utils-1.0.79-4.oe2203.x86_64
/etc/ld.so.conf.d/scsi-target-utils-x86_64.conf
/etc/sysconfig/tgtd
/etc/tgt
/etc/tgt/conf.d
/etc/tgt/conf.d/sample.conf
/etc/tgt/targets.conf
/etc/tgt/tgtd.conf
/usr/lib/systemd/system/tgtd.service
/usr/sbin/tgt-admin
/usr/sbin/tgt-setup-lun
/usr/sbin/tgtadm
/usr/sbin/tgtd
/usr/sbin/tgtimg
/usr/share/doc/scsi-target-utils
/usr/share/doc/scsi-target-utils/README
/usr/share/doc/scsi-target-utils/README.iscsi
/usr/share/doc/scsi-target-utils/README.iser
/usr/share/doc/scsi-target-utils/README.lu_configuration
/usr/share/doc/scsi-target-utils/README.mmc
/usr/share/doc/scsi-target-utils/README.ssc
```

Start the target service and enable it to automatically start upon system startup.

```
systemctl start tgtd  
systemctl enable tgtd
```

```
[root@ecs-san ~]# systemctl start tgtd  
[root@ecs-san ~]# systemctl enable tgtd  
Created symlink /etc/systemd/system/multi-user.target.wants/tgtd.service → /usr/lib/systemd/system/tgtd.service.  
[root@ecs-san ~]#
```

Check the service status and open ports.

```
systemctl status tgtd  
netstat -tnlp | grep 3260
```

```
[root@ecs-san ~]# systemctl status tgtd  
● tgtd.service - tgtd iSCSI target daemon  
   Loaded: loaded (/usr/lib/systemd/system/tgtd.service; enabled; vendor preset: disabled)  
   Active: active (running) since Thu 2023-04-27 10:12:10 CST; 31s ago  
     Main PID: 1876 (tgtd)  
        Tasks: 1 (limit: 21624)  
       Memory: 5.1M  
      CGroup: /system.slice/tgtd.service  
           └─1876 /usr/sbin/tgtd -f  
  
Apr 27 10:12:05 ecs-san.novalocal systemd[1]: Starting tgtd iSCSI target daemon...  
Apr 27 10:12:05 ecs-san.novalocal tgtd[1876]: tgtd: iser_ib_init(3431) Failed to initialize RDMA; load kernel modules?  
Apr 27 10:12:05 ecs-san.novalocal tgtd[1876]: tgtd: work_timer_start(146) use timer_fd based scheduler  
Apr 27 10:12:05 ecs-san.novalocal tgtd[1876]: tgtd: bs_init(387) use signalfd notification  
Apr 27 10:12:10 ecs-san.novalocal systemd[1]: Started tgtd iSCSI target daemon.  
[root@ecs-san ~]# netstat -tnlp | grep 3260  
tcp        0      0 0.0.0.0:3260          0.0.0.0:*          LISTEN      1876/tgtd  
tcp6       0      0 ::::3260            ::::*           LISTEN      1876/tgtd  
[root@ecs-san ~]#
```

### Step 3 Create a target using the **tgtadm** command.

On **ecs-san**, run the following command to create a target:

```
tgtadm --lld iscsi --mode target --op new --tid 1 --targetname iqn.2023-04.com.openeuler:target1
```

View information about the created target. This example uses short options, while other examples in this lab use long options. Use whichever format as you prefer.

```
tgtadm -L iscsi -m target -o show
```

```
[root@ecs-san ~]# tgtadm -L iscsi -m target -o show
Target 1: iqn.2023-04.com.openeuler:target1
  System information:
    Driver: iscsi
    State: ready
  I_T nexus information:
    LUN information:
      LUN: 0
        Type: controller
        SCSI ID: IET      00010000
        SCSI SN: beaf10
        Size: 0 MB, Block size: 1
        Online: Yes
        Removable media: No
        Prevent removal: No
        Readonly: No
        SWP: No
        Thin-provisioning: No
        Backing store type: null
        Backing store path: None
        Backing store flags:
  Account information:
  ACL information:
[root@ecs-san ~]#
```

In the information about the created target, you can see that a logical unit with logical unit number (LUN) 0 is automatically created. LUN 0 is created for each target and acts as the controller. **Backing store type** is **null**, indicating that the logical unit has no underlying logical device.

Add a logical unit for the target. A data drive has been added during **ecs-san** creation. Check the location of the drive.

```
fdisk -l
```

```
[root@ecs-san ~]# fdisk -l
Disk /dev/vda: 40 GiB, 42949672960 bytes, 83886080 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: 0xba2c5c05

Device      Boot   Start     End   Sectors  Size Id Type
/dev/vda1    *       2048  2099199  2097152   1G 83 Linux
/dev/vda2        2099200 83886079 81786880  39G 8e Linux LVM

Disk /dev/mapper/openeuler-root: 35 GiB, 37576769536 bytes, 73392128 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

Disk /dev/mapper/openeuler-swap: 4 GiB, 4294967296 bytes, 8388608 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

Disk /dev/vdb: 10 GiB, 10737418240 bytes, 20971520 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
[root@ecs-san ~]# █
```

A 10 GB drive **/dev/vdb** is displayed. You can either add the entire drive as a logical unit to **target1**, or partition the drive and add a partition as a logical unit.

```
tgtadm --lld iscsi --mode logicalunit --op new --tid 1 --lun 1 --backing-store /dev/vdb
```

Check the target information.

```
tgtadm --lld iscsi --mode target --op show
```

```
[root@ecs-san ~]# tgtadm --lld iscsi --mode logicalunit --op new --tid 1 --lun 1 --Backing-store /dev/vdb
[root@ecs-san ~]# tgtadm --lld iscsi --mode target --op show
Target 1: iqn.2023-04.com.openeuler:target1
  System information:
    Driver: iscsi
    State: ready
  I_T nexus information:
  LUN information:
    LUN: 0
      Type: controller
      SCSI ID: IET      00010000
      SCSI SN: beaf10
      Size: 0 MB, Block size: 1
      Online: Yes
      Removable media: No
      Prevent removal: No
      Readonly: No
      SWP: No
      Thin-provisioning: No
      Backing store type: null
      Backing store path: None
      Backing store flags:
    LUN: 1
      Type: disk
      SCSI ID: IET      00010001
      SCSI SN: beaf11
      Size: 10737 MB, Block size: 512
      Online: Yes
      Removable media: No
      Prevent removal: No
      Readonly: No
      SWP: No
      Thin-provisioning: No
      Backing store type: rdwr
      Backing store path: /dev/vdb
      Backing store flags:
  Account information:
  ACL information:
```

You can see that LUN 1 has been added to **target1**. LUN 1 is a readable and writable (**rdwr**) device of the **disk** type and has logical device **/dev/vdb**. However, **Account information** and **ACL information** is still empty, indicating that the target is not shared.

#### Step 4 Configure sharing.

Bind an IP address to **target1** to allow the IP address to connect to **target1**. The following command allows **Client01** to connect to **target1**.

```
tgtadm --lld iscsi --mode target --op bind --tid 1 --initiator-address 192.168.1.21
```

Check the target information again.

```
tgtadm --lld iscsi --mode target --op show
```

```
[root@ecs-san ~]# tgtadm --lld iscsi --mode target --op bind --tid 1 --initiator-address 192.168.1.21
[root@ecs-san ~]# tgtadm --lld iscsi --mode target --op show
Target 1: iqn.2023-04.com.openeuler:target1
    System information:
        Driver: iscsi
        State: ready
    I_T nexus information:
    LUN information:
        LUN: 0
            Type: controller
            SCSI ID: IET      00010000
            SCSI SN: beaf10
            Size: 0 MB, Block size: 1
            Online: Yes
            Removable media: No
            Prevent removal: No
            Readonly: No
            SWP: No
            Thin-provisioning: No
            Backing store type: null
            Backing store path: None
            Backing store flags:
        LUN: 1
            Type: disk
            SCSI ID: IET      00010001
            SCSI SN: beaf11
            Size: 10737 MB, Block size: 512
            Online: Yes
            Removable media: No
            Prevent removal: No
            Readonly: No
            SWP: No
            Thin-provisioning: No
            Backing store type: rdwr
            Backing store path: /dev/vdb
            Backing store flags:
    Account information:
        ACL information:
            192.168.1.21
```

ACL information of target1 becomes 192.168.1.21.

- Question: Can you allow different IP addresses to access distinct LUNs by binding LUNs of the same target to different IP addresses?
- Answer: No. IP address-based sharing is at the target level. The bound IP address can access all LUNs of **target1**.

## Step 5 Configure the connection on the client.

Log in to **Client01** and install iscsi-initiator-utils.

```
yum -y install iscsi-initiator-utils
```

```
Last metadata expiration check: 0:59:03 ago on Thu 27 Apr 2023 10:32:11 AM CST.
Dependencies resolved.
=====
Package           Architecture Version       Repository
=====
Installing:
open-iscsi          x86_64     2.1.5-7.oe2203   0S
Installing dependencies:
open-libsns          x86_64     0.101-1.oe2203   0S
Installing weak dependencies:
open-libsns-help      noarch    0.101-1.oe2203   0S
=====
Transaction Summary
=====
Install 3 Packages
=====
Total download size: 460 k
Installed size: 2.0 M
Downloaded Packages:
(1/3): open-iscsi-2.1.5-7.oe2203.x86_64.rpm                                181 kB/s |  24 kB
(2/3): open-libsns-0.101-1.oe2203.noarch.rpm                               38 kB/s | 134 kB
(3/3): open-libsns-help-0.101-1.oe2203.x86_64.rpm                           Connection timeout after 2
                                                               300 kB/s |  15 kB/s | 460 kB
=====
Total
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing:
  Installing : open-libsns-help-0.101-1.oe2203.noarch
  Installing : open-iscsi-2.1.5-7.oe2203.x86_64
  Running scriptlet: open-libsns-0.101-1.oe2203.x86_64
  Installing : open-iscsi-2.1.5-7.oe2203.x86_64
  Running scriptlet: open-iscsi-2.1.5-7.oe2203.x86_64
  Verifying   : open-iscsi-2.1.5-7.oe2203.x86_64
  Verifying   : open-libsns-0.101-1.oe2203.x86_64
  Verifying   : open-libsns-help-0.101-1.oe2203.noarch
=====
Installed:
  open-iscsi-2.1.5-7.oe2203.x86_64
  open-libsns-0.101-1.oe2203.noarch
=====
Complete!
```

Start the iscsid service.

```
systemctl start iscsid
```

```
systemctl status iscsid  
systemctl enable iscsid
```

```
[root@Client01 ~]# systemctl start iscsid  
[root@Client01 ~]# systemctl status iscsid  
● iscsid.service - Open-iSCSI  
  Loaded: loaded (/usr/lib/systemd/system/iscsid.service; disabled; vendor preset: disabled)  
  Active: active (running) since Thu 2023-04-27 11:25:40 CST; 15min ago  
TriggeredBy: ● iscsid.socket  
    Docs: man:iscsid(8)  
          man:iscsiadm(8)  
 Main PID: 1889 (iscsid)  
   Status: "Ready to process requests"  
     Tasks: 1 (limit: 21624)  
    Memory: 3.0M  
   CGroup: /system.slice/iscsid.service  
           └─1889 /sbin/iscsid -f  
  
Apr 27 11:25:40 Client01 systemd[1]: Starting Open-iSCSI...  
Apr 27 11:25:40 Client01 systemd[1]: Started Open-iSCSI.
```

```
[root@Client01 ~]# systemctl enable iscsid  
Created symlink /etc/systemd/system/multi-user.target.wants/iscsid.service → /usr/lib/systemd/system/iscsid.service.  
[root@Client01 ~]#
```

## Step 6 Discover and connect to the target.

On **Client01**, run the following command to discover the target:

```
iscsiadm -m discovery -t sendtargets -p 192.168.1.3
```

```
[root@Client01 ~]# iscsiadm -m discovery -t sendtargets -p 192.168.1.3  
192.168.1.3:3260,1 iqn.2023-04.com.openeuler:target1  
[root@Client01 ~]#
```

## Step 7 Associate and disassociate the target.

Create a session between the initiator and target by associating the target. This session enables the initiator to view, access, and operate SCSI devices of the target.

On **Client01**, run the following command to associate all discovered targets:

```
iscsiadm -m node -L all
```

```
[root@Client01 ~]# iscsiadm -m node -L all  
Logging in to [iface: default, target: iqn.2023-04.com.openeuler:target1, portal: 192.168.1.3,3260]  
Login to [iface: default, target: iqn.2023-04.com.openeuler:target1, portal: 192.168.1.3,3260] successful.  
[root@Client01 ~]#
```

Run the **fdisk -l** command to check the drive list.

```
[root@Client01 ~]# fdisk -l
Disk /dev/vda: 40 GiB, 42949672960 bytes, 83886080 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: 0xba2c5c05

Device      Boot   Start     End   Sectors  Size Id Type
/dev/vda1    *       2048  2099199  2097152   1G 83 Linux
/dev/vda2        2099200 83886079 81786880  39G 8e Linux LVM

Disk /dev/mapper/openeuler-root: 35 GiB, 37576769536 bytes, 73392128 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

Disk /dev/mapper/openeuler-swap: 4 GiB, 4294967296 bytes, 8388608 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

Disk /dev/sda: 10 GiB, 10737418240 bytes, 20971520 sectors
Disk model: VIRTUAL-DISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
[root@Client01 ~]#
```

On **ecs-san**, run the following command to check the connection status:

```
tgtadm --lld iscsi --op show --mode target
```

```
[root@ecs-san ~]# tgtadm --lld iscsi --mode target --op show
Target 1: iqn.2023-04.com.openeuler:target1
  System information:
    Driver: iscsi
    State: ready
  I_T nexus information:
    I_T nexus: 1
      Initiator: iqn.2012-01.com.openeuler:f8b9d3b2b8f alias: Client01
      Connection: 0
        IP Address: 192.168.1.21
  LUN information:
    LUN: 0
      Type: controller
      SCSI ID: IET      00010000
      SCSI SN: beaf10
      Size: 0 MB, Block size: 1
      Online: Yes
      Removable media: No
      Prevent removal: No
      Readonly: No
      SWP: No
      Thin-provisioning: No
      Backing store type: null
      Backing store path: None
      Backing store flags:
    LUN: 1
      Type: disk
      SCSI ID: IET      00010001
      SCSI SN: beaf11
      Size: 10737 MB, Block size: 512
      Online: Yes
      Removable media: No
      Prevent removal: No
      Readonly: No
      SWP: No
      Thin-provisioning: No
      Backing store type: rdwr
      Backing store path: /dev/vdb
      Backing store flags:
  Account information:
  ACL information:
    192.168.1.21
[root@ecs-san ~]#
```

**I\_T nexus information** shows information about connections between initiators and LUNs of the target. **nexus** is the number of association times.

Run the following command to disassociate the target:

```
iscsiadm -m node -T iqn.2023-04.com.openeuler:target1 -u
```

```
[root@Client01 ~]# iscsiadm -m node -T iqn.2023-04.com.openeuler:target1 -u
Logging out of session [sid: 2, target: iqn.2023-04.com.openeuler:target1, portal: 192.168.1.3,3260]
Logout of [sid: 2, target: iqn.2023-04.com.openeuler:target1, portal: 192.168.1.3,3260] successful.
[root@Client01 ~]#
```

```
[root@Client01 ~]# fdisk -l
Disk /dev/vda: 40 GiB, 42949672960 bytes, 83886080 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: 0xba2c5c05

Device      Boot   Start     End   Sectors  Size Id Type
/dev/vda1    *       2048  2099199  2097152   1G 83 Linux
/dev/vda2        2099200 83886079 81786880  39G 8e Linux LVM

Disk /dev/mapper/openeuler-root: 35 GiB, 37576769536 bytes, 73392128 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

Disk /dev/mapper/openeuler-swap: 4 GiB, 4294967296 bytes, 8388608 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
[root@Client01 ~]# █
```

Run the following command to associate target 1 again:

```
iscsiadm -m node -T iqn.2023-04.com.openeuler:target1 -p 192.168.1.3 -l
```

```
[root@Client01 ~]# iscsiadm -m node -T iqn.2023-04.com.openeuler:target1 -p 192.168.1.3 -l
Logging in to [iface: default, target: iqn.2023-04.com.openeuler:target1, portal: 192.168.1.3,3260]
Login to [iface: default, target: iqn.2023-04.com.openeuler:target1, portal: 192.168.1.3,3260] successful.
[root@Client01 ~]# █
```

Run the **fdisk -l** command to check the drive list again.

```
[root@Client01 ~]# fdisk -l
Disk /dev/vda: 40 GiB, 42949672960 bytes, 83886080 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: 0xba2c5c05

Device      Boot   Start     End   Sectors Size Id Type
/dev/vda1    *       2048 2099199 2097152   1G 83 Linux
/dev/vda2        2099200 83886079 81786880 39G 8e Linux LVM

Disk /dev/mapper/openeuler-root: 35 GiB, 37576769536 bytes, 73392128 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

Disk /dev/mapper/openeuler-swap: 4 GiB, 4294967296 bytes, 8388608 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

Disk /dev/sda: 10 GiB, 10737418240 bytes, 20971520 sectors
Disk model: VIRTUAL-DISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
[root@Client01 ~]#
```

Check the connection status.

```
iscsiadm -m session
```

```
[root@Client01 ~]# iscsiadm -m session
tcp: [5] 192.168.1.3:3260,1 iqn.2023-04.com.openeuler:target1 (non-flash)
[root@Client01 ~]#
```

## Step 8 Format the drive.

On **Client01**, check the drive name, format the drive, and write a file to it.

```
fdisk -l
```

```
[root@Client01 ~]# fdisk -l
Disk /dev/vda: 40 GiB, 42949672960 bytes, 83886080 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: 0xba2c5c05

Device      Boot   Start     End   Sectors  Size Type
/dev/vda1    *       2048  2099199  2097152   1G 83 Linux
/dev/vda2        2099200 83886079 81786880  39G 8e Linux LVM

Disk /dev/mapper/openeuler-root: 35 GiB, 37576769536 bytes, 73392128 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

Disk /dev/mapper/openeuler-swap: 4 GiB, 4294967296 bytes, 8388608 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

Disk /dev/sda: 10 GiB, 10737418240 bytes, 20971520 sectors
Disk model: VIRTUAL-DISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

```
mkfs.ext4 /dev/sda
```

```
[root@Client01 ~]# mkfs.ext4 /dev/sda
mke2fs 1.46.4 (18-Aug-2021)
Found a dos partition table in /dev/sda
Proceed anyway? (y,N) y
Creating filesystem with 2621440 4k blocks and 655360 inodes
Filesystem UUID: 9d47f100-f312-45a5-bf18-ffd940a2973a
Superblock backups stored on blocks:
            32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

Check the drive format.

```
blkid /dev/sda
```

```
[root@Client01 ~]# blkid /dev/sda
/dev/sda: UUID="9d47f100-f312-45a5-bf18-ffd940a2973a" BLOCK_SIZE="4096" TYPE="ext4"
[root@Client01 ~]#
```

- Question: If you format a server drive through a client-side operation, will the changes be reflected on the server as well?

Answer: Yes. The drive format information on ecs-san is as follows.

```
blkid /dev/vdb
```

```
[root@ecs-san ~]# blkid /dev/vdb
/dev/vdb: UUID="9d47f100-f312-45a5-bf18-ffd940a2973a" BLOCK_SIZE="4096" TYPE="ext4"
[root@ecs-san ~]#
```

The drive format is **ext4**, indicating that the drive on the SAN is formatted.

## Step 9 Mount and use the drive.

On **Client01**, and create a directory.

```
mkdir /mnt/san
```

Mount the drive to the directory.

```
mount /dev/sda /mnt/san  
df -h
```

```
[root@Client01 ~]# mkdir /mnt/san  
[root@Client01 ~]# mount /dev/sda /mnt/san  
[root@Client01 ~]# df -h  
Filesystem           Size   Used  Avail Use% Mounted on  
devtmpfs             1.7G    0     1.7G  0% /dev  
tmpfs               1.7G    0     1.7G  0% /dev/shm  
tmpfs               683M   8.6M  674M  2% /run  
tmpfs               4.0M    0     4.0M  0% /sys/fs/cgroup  
/dev/mapper/openeuler-root  35G   2.5G  30G  8% /  
tmpfs               1.7G   32K   1.7G  1% /tmp  
/dev/vda1            974M   87M  820M 10% /boot  
/dev/sda             9.8G   24K   9.3G  1% /mnt/san  
[root@Client01 ~]#
```

In **/mnt/san**, create file **test.txt** and write **This is from Client01** into it.

```
[root@Client01 ~]# cd /mnt/san  
[root@Client01 san]# echo "This is from Client01" > test.txt  
[root@Client01 san]# cat test.txt  
This is from Client01  
[root@Client01 san]#
```

On **ecs-san**, bind the IP address of **Client02** to **target1**.

```
tgtadm --lld iscsi --mode target --op bind --tid 1 --initiator-address 192.168.1.22
```

```
[root@ecs-san ~]# tgtadm --lld iscsi --mode target --op bind --tid 1 --initiator-address 192.168.1.22
[root@ecs-san ~]# tgtadm --lld iscsi --op show --mode target
Target 1: iqn.2023-04.com.openeuler:target1
    System information:
        Driver: iscsi
        State: ready
    I_T nexus information:
        I_T nexus: 1
            Initiator: iqn.2012-01.com.openeuler:f8b9d3b2b8f alias: Client01
            Connection: 0
            IP Address: 192.168.1.21
    LUN information:
        LUN: 0
            Type: controller
            SCSI ID: IET      00010000
            SCSI SN: beaf10
            Size: 0 MB, Block size: 1
            Online: Yes
            Removable media: No
            Prevent removal: No
            Readonly: No
            SWP: No
            Thin-provisioning: No
            Backing store type: null
            Backing store path: None
            Backing store flags:
        LUN: 1
            Type: disk
            SCSI ID: IET      00010001
            SCSI SN: beaf11
            Size: 10737 MB, Block size: 512
            Online: Yes
            Removable media: No
            Prevent removal: No
            Readonly: No
            SWP: No
            Thin-provisioning: No
            Backing store type: rdwr
            Backing store path: /dev/vdb
            Backing store flags:
    Account information:
        ACL information:
            192.168.1.21
            192.168.1.22
[root@ecs-san ~]#
```

Log in to **Client02** and repeat steps 5 to 7 to connect to **target1**.

```
[root@Client02 ~]# iscsidadm -m session
tcp: [1] 192.168.1.3:3260,1 iqn.2023-04.com.openeuler:target1 (non-flash)
[root@Client02 ~]#
```

On **ecs-san**, run the following command to check the connection status:

`tgtadm --lld iscsi --op show --mode target`

Two **I\_T nexus** records are displayed, each corresponding to a distinct initiator.

```
[root@ecs-san ~]# tgtadm --lld iscsi --op show --mode target
Target 1: iqn.2023-04.com.openeuler:target1
  System information:
    Driver: iscsi
    State: ready
  I_T nexus information:
    I_T nexus: 1
      Initiator: iqn.2012-01.com.openeuler:f8b9d3b2b8f alias: Client01
      Connection: 0
      IP Address: 192.168.1.21
    I_T nexus: 2
      Initiator: iqn.2012-01.com.openeuler:7b252b84f34 alias: Client02
      Connection: 0
      IP Address: 192.168.1.22
  LUN information:
    LUN: 0
      Type: controller
      SCSI ID: IET 00010000
      SCSI SN: beaf10
      Size: 0 MB, Block size: 1
      Online: Yes
      Removable media: No
      Prevent removal: No
      Readonly: No
      SWP: No
      Thin-provisioning: No
      Backing store type: null
      Backing store path: None
      Backing store flags:
    LUN: 1
      Type: disk
      SCSI ID: IET 00010001
      SCSI SN: beaf11
      Size: 10737 MB, Block size: 512
      Online: Yes
      Removable media: No
      Prevent removal: No
      Readonly: No
      SWP: No
      Thin-provisioning: No
      Backing store type: rdwr
      Backing store path: /dev/vdb
      Backing store flags:
  Account information:
  ACL information:
    192.168.1.21
    192.168.1.22
[root@ecs-san ~]#
```

On Client02, check the drive information.

```
lsblk
```

```
[root@Client02 ~]# lsblk
NAME           MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda            8:0    0 10G  0 disk
vda            252:0   0 40G  0 disk
└─vda1         252:1   0  1G  0 part /boot
└─vda2         252:2   0 39G  0 part
  ├─openeuler-root 253:0   0 35G  0 lvm  /
  └─openeuler-swap 253:1   0  4G  0 lvm  [SWAP]
[root@Client02 ~]#
```

Create a directory.

```
mkdir /mnt/san
```

Mount drive **/dev/sda** to the directory.

```
mount /dev/sda /mnt/san
```

```
[root@Client02 ~]# mkdir /mnt/san
[root@Client02 ~]# mount /dev/sda /mnt/san
[root@Client02 ~]# df -h
Filesystem           Size   Used  Avail Use% Mounted on
devtmpfs             1.7G    0     1.7G  0% /dev
tmpfs                1.7G    0     1.7G  0% /dev/shm
tmpfs                683M   8.6M  674M  2% /run
tmpfs                4.0M    0     4.0M  0% /sys/fs/cgroup
/dev/mapper/openeuler-root  35G   2.5G  30G  8% /
tmpfs                1.7G   32K  1.7G  1% /tmp
/dev/vda1              974M   87M  820M 10% /boot
/dev/sda               9.8G   28K  9.3G  1% /mnt/san
[root@Client02 ~]#
```

Check files in the mounted directory. The file created by **Client01** exists.

```
[root@Client02 ~]# ls /mnt/san
lost+found  test.txt
[root@Client02 ~]#
```

On Client02, go to /mnt/san/, create file test1.txt, and write This is from Client02 to it.

```
[root@Client02 ~]# cd /mnt/san
[root@Client02 san]# echo "This is from Client02" > test1.txt
[root@Client02 san]# cat test1.txt
This is from Client02
[root@Client02 san]#
```

On Client01, and check the content of /mnt/san. You can see that data is not synchronized.

```
[root@Client01 san]# ls
lost+found  test.txt
[root@Client01 san]#
```

On Client01, create file **test2.txt**.

```
echo "This is from Client01 again" > test2.txt
cat test2.txt
```

```
[root@Client01 san]# echo "This is from Client01 again" > test2.txt
[root@Client01 san]# cat test2.txt
This is from Client01 again
```

On Client02, unmount /mnt/san and mount it again. You can see that **test1.txt** is lost.

```
[root@Client02 ~]# umount /mnt/san
[root@Client02 ~]# mount /dev/sda /mnt/san/
[root@Client02 ~]# ls /mnt/san
lost+found  test2.txt  test.txt
[root@Client02 ~]#
```

The logical storage device is accessed by multiple hosts simultaneously. As a result, data may conflict and cannot be synchronized in a timely manner, causing data loss.

- Question: When multiple hosts access a logical storage device simultaneously, how do you keep data synchronized and prevent data loss?

Answer: Configure a cluster or distributed file system for iSCSI to ensure data synchronization and loss prevention.

#### Step 10 Unmount the drive and disconnect the clients from the target.

On Client01, unmount /dev/sda.

```
umount /mnt/san
df -h
```

```
[root@Client01 ~]# umount /mnt/san
[root@Client01 ~]# df -h
Filesystem           Size   Used  Avail Use% Mounted on
/devtmpfs            1.7G    0     1.7G  0% /dev
tmpfs               1.7G    0     1.7G  0% /dev/shm
tmpfs               683M   8.6M  674M  2% /run
tmpfs               4.0M    0     4.0M  0% /sys/fs/cgroup
/dev/mapper/openeuler-root  35G   2.5G  30G  8% /
tmpfs               1.7G   32K   1.7G  1% /tmp
/dev/vda1            974M   87M   820M  10% /boot
[root@Client01 ~]#
```

Disconnect from **target1**.

```
iscsiadm -m node -T iqn.2023-04.com.openeuler:target1 -u
iscsiadm -m session
```

```
[root@Client01 ~]# iscsiadm -m node -T iqn.2023-04.com.openeuler:target1 -u
Logging out of session [sid: 5, target: iqn.2023-04.com.openeuler:target1, portal: 192.168.1.3,3260]
Logout of [sid: 5, target: iqn.2023-04.com.openeuler:target1, portal: 192.168.1.3,3260] successful.
[root@Client01 ~]# iscsiadm -m session
iscsiadm: No active sessions.
[root@Client01 ~]#
```

On Client02, repeat the operations to unmount the drive and disconnect from **target1**.

```
[root@Client02 ~]# iscsiadm -m node -T iqn.2023-04.com.openeuler:target1 -u
Logging out of session [sid: 1, target: iqn.2023-04.com.openeuler:target1, portal: 192.168.1.3,3260]
Logout of [sid: 1, target: iqn.2023-04.com.openeuler:target1, portal: 192.168.1.3,3260] successful.
[root@Client02 ~]# iscsiadm -m session
iscsiadm: No active sessions.
[root@Client02 ~]#
```

## Step 11 Delete the target on the server.

On **ecs-san**, check the target information. No connection exists.

```
[root@ecs-san ~]# tgtadm --lld iscsi --op show --mode target
Target 1: iqn.2023-04.com.openeuler:target1
  System information:
    Driver: iscsi
    State: ready
  I_T nexus information:
  LUN information:
    LUN: 0
      Type: controller
      SCSI ID: IET 00010000
      SCSI SN: beaf10
      Size: 0 MB, Block size: 1
      Online: Yes
      Removable media: No
      Prevent removal: No
      Readonly: No
      SWP: No
      Thin-provisioning: No
      Backing store type: null
      Backing store path: None
      Backing store flags:
    LUN: 1
      Type: disk
      SCSI ID: IET 00010001
      SCSI SN: beaf11
      Size: 10737 MB, Block size: 512
      Online: Yes
      Removable media: No
      Prevent removal: No
      Readonly: No
      SWP: No
      Thin-provisioning: No
      Backing store type: rdwr
      Backing store path: /dev/vdb
      Backing store flags:
  Account information:
  ACL information:
    192.168.1.21
    192.168.1.22
```

Delete the connection addresses, logical unit, and **target1** in sequence.

```
tgtadm --lld iscsi --mode target --op unbind --tid 1 --initiator-address 192.168.1.21
tgtadm --lld iscsi --mode target --op unbind --tid 1 --initiator-address 192.168.1.22
```

```
[root@ecs-san ~]# tgtadm --lld iscsi --mode target --op unbind --tid 1 --initiator-address 192.168.1.21
[root@ecs-san ~]# tgtadm --lld iscsi --mode target --op unbind --tid 1 --initiator-address 192.168.1.22
[root@ecs-san ~]#
```

```
tgtadm --lld iscsi --mode logicalunit --op delete --tid 1 --lun 1
tgtadm --lld iscsi --mode target --op show
```

```
[root@ecs-san ~]# tgtadm --lld iscsi --mode logicalunit --op delete --tid 1 --lun 1
[root@ecs-san ~]#
[root@ecs-san ~]# tgtadm --lld iscsi --mode target --op show
Target 1: iqn.2023-04.com.openeuler:target1
  System information:
    Driver: iscsi
    State: ready
  I_T nexus information:
  LUN information:
    LUN: 0
      Type: controller
      SCSI ID: IET      00010000
      SCSI SN: beaf10
      Size: 0 MB, Block size: 1
      Online: Yes
      Removable media: No
      Prevent removal: No
      Readonly: No
      SWP: No
      Thin-provisioning: No
      Backing store type: null
      Backing store path: None
      Backing store flags:
    Account information:
    ACL information:
[root@ecs-san ~]#
```

Run the following command to delete **target1**:

```
tgtadm --lld iscsi --mode target --op delete --tid 1
```

```
[root@ecs-san ~]# tgtadm --lld iscsi --mode target --op delete --tid 1
[root@ecs-san ~]# tgtadm --lld iscsi --mode target --op show
[root@ecs-san ~]#
```

## Step 12 Create a target using the target configuration file.

Configurations set by the **tgtadm** command are saved in the memory and will be lost when the tgt service or system restarts. To make the configurations persistent, save them to the configuration file.

The default target configuration file is **/etc/tgt/targets.conf**.

On **ecs-san**, add the following content to the end of **/etc/tgt/targets.conf**, save the file, and exit.

```
<target iqn.2023-04.com.openeuler:target2>          # Start creation. Edit the target and IQN.
  backing-store /dev/vdb                                # Add a drive.
  initiator-address 192.168.1.0/24                      # Allow connections from subnet
  192.168.1.0/24.
</target>                                              # End target creation.
```

```
<target iqn.2023-04.com.openeuler:target2>
  backing-store /dev/vdb
  initiator-address 192.168.1.0/24
</target>
```

Restart the tgtd service and check the target information.

```
systemctl restart tgtd
tgtadm --lld iscsi --op show --mode target
```

```
[root@ecs-san ~]# tgtadm --lld iscsi --op show --mode target
Target 1: iqn.2023-04.com.openeuler:target2
  System information:
    Driver: iscsi
    State: ready
  I_T nexus information:
  LUN information:
    LUN: 0
      Type: controller
      SCSI ID: IET      00010000
      SCSI SN: beaf10
      Size: 0 MB, Block size: 1
      Online: Yes
      Removable media: No
      Prevent removal: No
      Readonly: No
      SWP: No
      Thin-provisioning: No
      Backing store type: null
      Backing store path: None
      Backing store flags:
    LUN: 1
      Type: disk
      SCSI ID: IET      00010001
      SCSI SN: beaf11
      Size: 10737 MB, Block size: 512
      Online: Yes
      Removable media: No
      Prevent removal: No
      Readonly: No
      SWP: No
      Thin-provisioning: No
      Backing store type: rdwr
      Backing store path: /dev/vdb
      Backing store flags:
  Account information:
  ACL information:
    192.168.1.0/24
[root@ecs-san ~]#
```

On Client01, connect to and mount the target. Then, test the connection.

```
[root@Client01 ~]# iscsidadm -m discovery -t sendtargets -p 192.168.1.3
192.168.1.3:3260,1 iqn.2023-04.com.openeuler:target2
[root@Client01 ~]# iscsidadm -m node -L all
Logging in to [iface: default, target: iqn.2023-04.com.openeuler:target2, portal: 192.168.1.3,3260]
Login to [iface: default, target: iqn.2023-04.com.openeuler:target2, portal: 192.168.1.3,3260] successful.
[root@Client01 ~]# mount /dev/sda  /mnt/san
[root@Client01 ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        1.7G   0   1.7G  0% /dev
tmpfs          1.7G   0   1.7G  0% /dev/shm
tmpfs         683M  8.6M 674M  2% /run
tmpfs          4.0M   0   4.0M  0% /sys/fs/cgroup
/dev/mapper/openeuler-root  35G  2.5G  30G  8% /
tmpfs          1.7G  32K  1.7G  1% /tmp
/dev/vda1       974M  87M  820M 10% /boot
/dev/sda        9.8G  36K  9.3G  1% /mnt/san
[root@Client01 ~]#
```

Step 13    Enable the client to automatically mount the SAN storage upon system startup.

By default, the storage mounting configuration is lost after a client is restarted, and manual remounting is required. Configuring automatic mounting at startup streamlines operations and enhances user experience.

On **Client01**, check whether the mounted directory still exists after the restart.

```
Authorized users only. All activities may be monitored and reported.
Last login: Thu ...

Welcome to 5.10.0-60.18.0.50.oe2203.x86_64

System information as of time: Thu Apr 27 05:54:36 PM CST 2023

System load: 0.36
Processes: 127
Memory used: 4.4%
Swap used: 0%
Usage On: 8%
IP address: 192.168.1.21
Users online: 1

[root@Client01 ~]# df -h
Filesystem           Size  Used Avail Use% Mounted on
devtmpfs              1.7G    0  1.7G  0% /dev
tmpfs                 1.7G    0  1.7G  0% /dev/shm
tmpfs                 683M  8.6M 674M  2% /run
tmpfs                 4.0M    0  4.0M  0% /sys/fs/cgroup
/dev/mapper/openeuler-root  35G  2.5G  30G  8% /
tmpfs                 1.7G  32K  1.7G  1% /tmp
/dev/vda1              974M  87M  820M 10% /boot
[root@Client01 ~]#
```

The mount point does not exist.

Check the drive information. The drive is not connected. Reconnect and log in to the target.

```
[root@Client01 ~]# fdisk -l
Disk /dev/vda: 40 GiB, 42949672960 bytes, 83886080 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: 0xba2c5c05

Device      Boot   Start     End   Sectors  Size Id Type
/dev/vda1    *    2048 2099199  2097152   1G 83 Linux
/dev/vda2        2099200 83886079 81786880   39G 8e Linux LVM

Disk /dev/mapper/openeuler-root: 35 GiB, 37576769536 bytes, 73392128 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

Disk /dev/mapper/openeuler-swap: 4 GiB, 4294967296 bytes, 8388608 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

[root@Client01 ~]# iscsadm -m discovery -t sendtargets -p 192.168.1.3
192.168.1.3:3260,1 iqn.2023-04.com.openeuler:target2
[root@Client01 ~]# iscsiadm -m node -L all
Logging in to [iface: default, target: iqn.2023-04.com.openeuler:target2, portal: 192.168.1.3,3260]
Login to [iface: default, target: iqn.2023-04.com.openeuler:target2, portal: 192.168.1.3,3260] successful.
[root@Client01 ~]# lsblk
NAME      MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
sda          8:0    0 10G  0 disk
vda        252:0    0 40G  0 disk
└─vda1     252:1    0   1G  0 part /boot
└─vda2     252:2    0 39G  0 part
  └─openeuler-root 253:0    0 35G  0 lvm /
  └─openeuler-swap 253:1    0   4G  0 lvm [SWAP]
[root@Client01 ~]#
```

Run the following command to enable automatic login upon system startup:

```
iscsiadm -m node -T iqn.2023-04.com.openeuler:target2 -p 192.168.1.3 --op update -n node.startup -v automatic
```

```
[root@Client01 ~]# iscsiadm -m node -T iqn.2023-04.com.openeuler:target2 -p 192.168.1.3 --op update -n node.startup -v automatic
```

Remount drive **/dev/sda** to mount point **/mnt/san**.

```
[root@Client01 ~]# mount /dev/sda /mnt/san
[root@Client01 ~]# df -h
Filesystem           Size  Used Avail Use% Mounted on
devtmpfs              1.7G    0  1.7G  0% /dev
tmpfs                 1.7G    0  1.7G  0% /dev/shm
tmpfs                 683M  8.6M 674M  2% /run
tmpfs                 4.0M    0  4.0M  0% /sys/fs/cgroup
/dev/mapper/openeuler-root  35G  2.5G 30G  8% /
tmpfs                 1.7G  32K  1.7G  1% /tmp
/dev/vda1              974M  87M 820M 10% /boot
/dev/sda                9.8G  36K  9.3G  1% /mnt/san
[root@Client01 ~]#
```

Add the following content to **/etc/fstab** and save the file:

```
/dev/sda /mnt/san ext4 defaults,_netdev 0 0
```

```
"/dev/mapper/openeuler-root /           ext4    defaults        1 1
UUID=ea26a7d-9b83-4a0d-b219-ae8a0191cd72 /boot      ext4    defaults        1 2
/dev/mapper/openeuler-swap none        swap    defaults        0 0
192.168.1.10:/data /mnt/data nfs defaults,_netdev 0 0
/dev/sda /mnt/san ext4 defaults,_netdev 0 0
```

Restart **Client01** again and check whether the drive is mounted.

```
[root@Client01 ~]# df -h
Filesystem           Size  Used Avail Use% Mounted on
devtmpfs              1.7G   0    1.7G  0% /dev
tmpfs                 1.7G   0    1.7G  0% /dev/shm
tmpfs                 683M  8.6M  674M  2% /run
tmpfs                 4.0M   0    4.0M  0% /sys/fs/cgroup
/dev/mapper/openeuler-root  35G  2.5G  30G  8% /
tmpfs                 1.7G  32K  1.7G  1% /tmp
/dev/vda1              974M  87M  820M 10% /boot
192.168.1.10:/data      35G  2.5G  30G  8% /mnt/data
/dev/sda                9.8G  36K  9.3G  1% /mnt/san
[root@Client01 ~]#
```

# 2 GlusterFS Distributed Storage

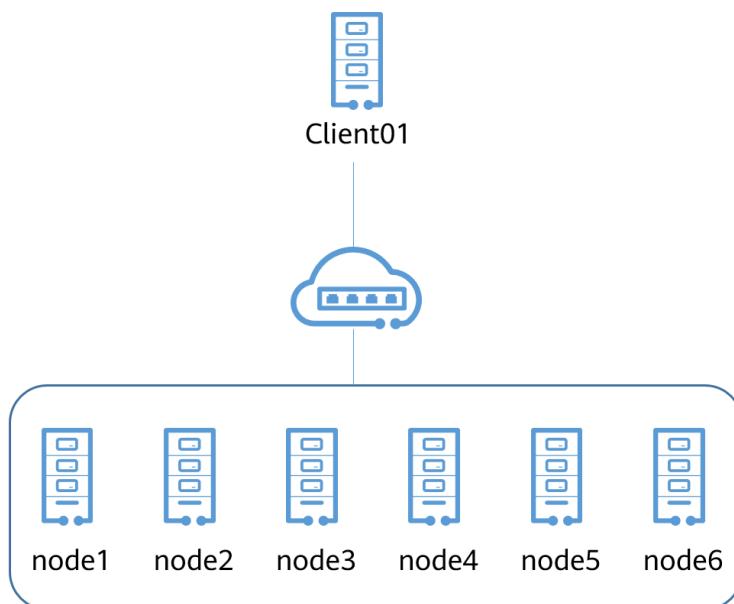
## 2.1 Introduction

In this lab, you will use ECSs as storage nodes to set up a GlusterFS cluster and create different types of volumes for client access. Lab operations include:

- GlusterFS server cluster setup
- GlusterFS client setup
- Creation and usage of different types of volumes
- Cluster and volume management

## 2.2 Networking

This lab involves seven ECSs. You can reuse **Client01** from the shared storage management lab and create six additional ECSs as storage nodes to form a GlusterFS distributed storage cluster. The following figure shows the topology.



ECS Name	IP Address	Drives	Remarks	Specifications
node1	192.168.1.31	/dev/vdb1 2G /dev/vdb2 2G /dev/vdb3 2G <b>/dev/vdb4 5G</b> /dev/vdb5 2G /dev/vdb6 2G	For the <b>/exp/vdb1</b> distributed volume For the <b>/exp/vdb2</b> replicated volume For the <b>/exp/vdb3</b> distributed replicated volume For the <b>/exp/vdb5</b> dispersed volume For the <b>/exp/vdb6</b> distributed dispersed volume	
node2	192.168.1.32	/dev/vdb1 2G /dev/vdb2 2G /dev/vdb3 2G <b>/dev/vdb4 5G</b> /dev/vdb5 2G /dev/vdb6 2G	For the <b>/exp/vdb1</b> distributed volume For the <b>/exp/vdb2</b> replicated volume For the <b>/exp/vdb3</b> distributed replicated volume For the <b>/exp/vdb5</b> dispersed volume For the <b>/exp/vdb6</b> distributed dispersed volume	OS: openEuler 22.03 LTS Password: Set a password as needed. Flavor: 1 vCPU   2 GB System drive: 40 GB Data drive: 11 GB
node3	192.168.1.33	/dev/vdb1 2G /dev/vdb2 2G /dev/vdb3 2G <b>/dev/vdb4 5G</b> /dev/vdb5 2G /dev/vdb6 2G	For the <b>/exp/vdb2</b> replicated volume For the <b>/exp/vdb3</b> distributed replicated volume For the <b>/exp/vdb5</b> dispersed volume For the <b>/exp/vdb6</b> distributed dispersed volume	Each ECS is bound to an EIP.
node4	192.168.1.34	/dev/vdb1 2G /dev/vdb2 2G /dev/vdb3 2G <b>/dev/vdb4 5G</b> /dev/vdb5 2G	For the <b>/exp/vdb1</b> test volume For the <b>/exp/vdb3</b> distributed replicated volume	

ECS Name	IP Address	Drives		Remarks	Specifications
		<code>/dev/vdb6</code> 2G		For the <code>/exp/vdb6</code> distributed dispersed volume	
node5	192.168.1.35	<code>/dev/vdb1</code>	2G	For the <code>/exp/vdb1</code> test volume	
		<code>/dev/vdb2</code>	2G	For the <code>/exp/vdb6</code> distributed dispersed volume	
		<code>/dev/vdb3</code>	2G	For the <code>/exp/vdb6</code> distributed dispersed volume	
		<code>/dev/vdb4</code>	5G	For the <code>/exp/vdb6</code> distributed dispersed volume	
		<code>/dev/vdb5</code>	2G	For the <code>/exp/vdb6</code> distributed dispersed volume	
		<code>/dev/vdb6</code>	2G	For the <code>/exp/vdb6</code> distributed dispersed volume	
node6	192.168.1.36	<code>/dev/vdb1</code>	2G	For the <code>/exp/vdb6</code> distributed dispersed volume	
		<code>/dev/vdb2</code>	2G	For the <code>/exp/vdb6</code> distributed dispersed volume	
		<code>/dev/vdb3</code>	2G	For the <code>/exp/vdb6</code> distributed dispersed volume	
		<code>/dev/vdb4</code>	5G	For the <code>/exp/vdb6</code> distributed dispersed volume	
		<code>/dev/vdb5</code>	2G	For the <code>/exp/vdb6</code> distributed dispersed volume	
		<code>/dev/vdb6</code>	2G	For the <code>/exp/vdb6</code> distributed dispersed volume	
Client01	192.168.1.21			Reused	OS: openEuler 22.03 LTS Password: Set a password as needed. Flavor: 2 vCPUs   4 GB Drive: 40 GB Each ECS is bound to an EIP.

## 2.3 Procedure

Step 1 Log in to Huawei Cloud and create six ECSs based on the resource information.

Step 2 Partition the data drive.

Log in to **node1** and check the drive status.

```
[root@node1 ~]# fdisk -l
Disk /dev/vda: 40 GiB, 42949672960 bytes, 83886080 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: 0xba2c5c05

Device      Boot   Start     End   Sectors  Size Id Type
/dev/vda1    *      2048  2099199  2097152   1G 83 Linux
/dev/vda2        2099200 83886079 81786880  39G 8e Linux LVM

Disk /dev/mapper/openeuler-root: 35 GiB, 37576769536 bytes, 73392128 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

Disk /dev/mapper/openeuler-swap: 4 GiB, 4294967296 bytes, 8388608 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

Disk /dev/vdb: 11 GiB, 11811160064 bytes, 23068672 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: 0x9ecfca81
[root@node1 ~]#
```

Partition drive **/dev/vdb** into **/dev/vdb1**, **/dev/vdb2**, **/dev/vdb3**, **/dev/vdb4**, **/dev/vdb5** and **/dev/vdb6**. **/dev/vdb1**, **/dev/vdb2**, and **/dev/vdb3** are primary partitions. **/dev/vdb5** and **/dev/vdb6** are 2 GB logical partitions. **/dev/vdb4** is a 5 GB extended partition.

```
Device      Boot   Start     End   Sectors  Size Id Type
/dev/vdb1        2048  4196351  4194304   2G 83 Linux
/dev/vdb2        4196352  8390655  4194304   2G 83 Linux
/dev/vdb3        8390656 12584959  4194304   2G 83 Linux
/dev/vdb4        12584960 23068671 10483712   5G  5 Extended
/dev/vdb5        12587008 16781311  4194304   2G 83 Linux
/dev/vdb6        16783360 20977663  4194304   2G 83 Linux

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

[root@node1 ~]#
[root@node1 ~]#
[root@node1 ~]# lsblk
NAME           MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
vda            252:0    0  40G  0 disk
└─vda1          252:1    0   1G  0 part /boot
└─vda2          252:2    0   39G  0 part
  └─openeuler-root 253:0    0   35G  0 lvm /
  └─openeuler-swap 253:1    0   4G  0 lvm [SWAP]
vdb            252:16   0 11G  0 disk
└─vdb1          252:17   0   2G  0 part
└─vdb2          252:18   0   2G  0 part
└─vdb3          252:19   0   2G  0 part
└─vdb4          252:20   0   1K  0 part
└─vdb5          252:21   0   2G  0 part
└─vdb6          252:22   0   2G  0 part
```

Format the created partitions.

```
mkfs.xfs /dev/vdb1  
mkfs.xfs /dev/vdb2  
mkfs.xfs /dev/vdb3  
mkfs.xfs /dev/vdb5  
mkfs.xfs /dev/vdb6
```

After the formatting is complete, check the drive types.

```
[root@node1 ~]# blkid /dev/vdb5  
/dev/vdb5: UUID="b383ddfc-8df6-4010-a95b-a05ce42b2da1" BLOCK_SIZE="512" TYPE="xfs" PARTUUID="9ecfca81-05"  
[root@node1 ~]# blkid /dev/vdb6  
/dev/vdb6: UUID="f2392337-b154-49a6-b93d-d9ba1d6f1050" BLOCK_SIZE="512" TYPE="xfs" PARTUUID="9ecfca81-06"  
[root@node1 ~]# blkid /dev/vdb1 /dev/vdb2 /dev/vdb3 /dev/vdb5 dev/vdb6  
/dev/vdb1: UUID="9442a34f-5828-492d-9832-903fb236eb9e" BLOCK_SIZE="512" TYPE="xfs" PARTUUID="9ecfca81-01"  
/dev/vdb2: UUID="c39c844b-0361-4f52-9d6b-47f638bb47bc" BLOCK_SIZE="512" TYPE="xfs" PARTUUID="9ecfca81-02"  
/dev/vdb3: UUID="0128027e-b21a-403b-8498-7e409e7131e5" BLOCK_SIZE="512" TYPE="xfs" PARTUUID="9ecfca81-03"
```

### Step 3 Mount the partitions.

Create mount points for the partitions.

```
mkdir -p /exp/vdb1 /exp/vdb2 /exp/vdb3 /exp/vdb5 /exp/vdb6
```

```
[root@node1 ~]# mkdir -p /exp/vdb1 /exp/vdb2 /exp/vdb3 /exp/vdb5 /exp/vdb6  
[root@node1 ~]#  
[root@node1 ~]# ls /exp  
vdb1 vdb2 vdb3 vdb4 vdb5 vdb6  
[root@node1 ~]#
```

Mount the partitions to the corresponding mount points.

```
mount /dev/vdb1 /exp/vdb1  
mount /dev/vdb2 /exp/vdb2  
mount /dev/vdb3 /exp/vdb3  
mount /dev/vdb5 /exp/vdb5  
mount /dev/vdb6 /exp/vdb6
```

```
[root@node1 ~]# mount /dev/vdb1 /exp/vdb1
[root@node1 ~]# mount /dev/vdb2 /exp/vdb2
[root@node1 ~]# mount /dev/vdb3 /exp/vdb3
[root@node1 ~]# mount /dev/vdb5 /exp/vdb5
[root@node1 ~]# mount /dev/vdb6 /exp/vdb6
[root@node1 ~]# df -h
Filesystem           Size   Used  Avail Use% Mounted on
devtmpfs             716M    0    716M   0% /dev
tmpfs                732M    0    732M   0% /dev/shm
tmpfs                293M   4.3M  289M   2% /run
tmpfs                4.0M    0    4.0M   0% /sys/fs/cgroup
/dev/mapper/openeuler-root  35G   2.5G  30G   8% /
tmpfs                732M   32K  732M   1% /tmp
/dev/vda1              974M  87M  820M  10% /boot
/dev/vdb1              2.0G  47M  2.0G   3% /exp/vdb1
/dev/vdb2              2.0G  47M  2.0G   3% /exp/vdb2
/dev/vdb3              2.0G  47M  2.0G   3% /exp/vdb3
/dev/vdb5              2.0G  47M  2.0G   3% /exp/vdb5
/dev/vdb6              2.0G  47M  2.0G   3% /exp/vdb6
[root@node1 ~]#
```

Edit **fstab** to enable automatic mounting upon system startup.

```
echo "/dev/vdb1 /exp/vdb1 xfs defaults 0 0" >> /etc/fstab
echo "/dev/vdb2 /exp/vdb2 xfs defaults 0 0" >> /etc/fstab
echo "/dev/vdb3 /exp/vdb3 xfs defaults 0 0" >> /etc/fstab
echo "/dev/vdb5 /exp/vdb5 xfs defaults 0 0" >> /etc/fstab
echo "/dev/vdb6 /exp/vdb6 xfs defaults 0 0" >> /etc/fstab
```

```
[root@node1 ~]# echo "/dev/vdb1 /exp/vdb1 xfs defaults 0 0" >> /etc/fstab
[root@node1 ~]# echo "/dev/vdb2 /exp/vdb2 xfs defaults 0 0" >> /etc/fstab
[root@node1 ~]# echo "/dev/vdb3 /exp/vdb3 xfs defaults 0 0" >> /etc/fstab
[root@node1 ~]# echo "/dev/vdb5 /exp/vdb5 xfs defaults 0 0" >> /etc/fstab
[root@node1 ~]# echo "/dev/vdb6 /exp/vdb6 xfs defaults 0 0" >> /etc/fstab
[root@node1 ~]# cat /etc/fstab

#
# /etc/fstab
# Created by anaconda on Mon Apr  3 08:21:21 2023
#
# Accessible filesystems, by reference, are maintained under '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
UUID=eae26a7d-9b83-4a0d-b219-ae8a0191cd72 /boot          ext4      defaults        1 1
/dev/mapper/openeuler-swap none                  swap      defaults        0 0
/dev/vdb1   /exp/vdb1  xfs defaults 0 0
/dev/vdb2   /exp/vdb2  xfs defaults 0 0
/dev/vdb3   /exp/vdb3  xfs defaults 0 0
/dev/vdb5   /exp/vdb5  xfs defaults 0 0
/dev/vdb6   /exp/vdb6  xfs defaults 0 0
[root@node1 ~]#
```

Create a subdirectory in each mount point as a brick of GlusterFS.

```
mkdir -p /exp/vdb1/brick /exp/vdb2/brick /exp/vdb3/brick /exp/vdb5/brick /exp/vdb6/brick
```

Step 4 Edit the **/etc/hosts** file to set host name mapping for each storage node.

```
echo "192.168.1.31 node1" >> /etc/hosts  
echo "192.168.1.32 node2" >> /etc/hosts  
echo "192.168.1.33 node3" >> /etc/hosts  
echo "192.168.1.34 node4" >> /etc/hosts  
echo "192.168.1.35 node5" >> /etc/hosts  
echo "192.168.1.36 node6" >> /etc/hosts
```

```
[root@node1 ~]# echo "192.168.1.31 node1" >> /etc/hosts  
echo "192.168.1.32 node2" >> /etc/hosts  
echo "192.168.1.33 node3" >> /etc/hosts  
echo "192.168.1.34 node4" >> /etc/hosts  
echo "192.168.1.35 node5" >> /etc/hosts  
echo "192.168.1.36 node6" >> /etc/hosts  
[root@node1 ~]# cat /etc/hosts  
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4  
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6  
192.168.1.31 node1  
192.168.1.32 node2  
192.168.1.33 node3  
192.168.1.34 node4  
192.168.1.35 node5  
192.168.1.36 node6  
[root@node1 ~]#
```

## Step 5 Install the GlusterFS server software package.

Run the following commands to install the GlusterFS server and start the service:

```
yum -y install glusterfs-server  
systemctl start glusterd  
systemctl enable glusterd
```

```
Installed:  
glusterfs-10.0-8.oe2203.x86_64      glusterfs-cli-10.0-8.oe2203.x86_64      glusterfs-client-xlators-10.0-8.oe2203.x86_64  
glusterfs-fuse-10.0-8.oe2203.x86_64    glusterfs-server-10.0-8.oe2203.x86_64    gperftools-libs-2.9.1-4.oe2203.x86_64  
libgfapi0-10.0-8.oe2203.x86_64       libgfchangelog-10.0-8.oe2203.x86_64    libgfRPC0-10.0-8.oe2203.x86_64  
libgfdxdr0-10.0-8.oe2203.x86_64      libglusterd0-10.0-8.oe2203.x86_64      libglusterfs0-10.0-8.oe2203.x86_64  
libunwind-2:1.5.0-2.oe2203.x86_64     liburing-0.7-2.oe2203.x86_64          python3-pyxattr-0.7.2-2.oe2203.x86_64  
  
Complete!  
[root@node1 ~]# systemctl start glusterd  
[root@node1 ~]# systemctl enable glusterd  
Created symlink /etc/systemd/system/multi-user.target.wants/glusterd.service → /usr/lib/systemd/system/glusterd.service.  
[root@node1 ~]#
```

## Step 6 Repeat steps 2 to 4 on node2, node3, node4, node5, and node6.

```
[root@node2 ~]# systemctl status glusterd
● glusterd.service - GlusterFS, a clustered file-system server
  Loaded: loaded (/usr/lib/systemd/system/glusterd.service; enabled; vendor preset: disabled)
  Active: active (running) since Mon 2023-05-08 09:23:33 CST; 36s ago
    Docs: man:glusterd(8)
    Main PID: 1903 (glusterd)
      Tasks: 24 (limit: 9154)
     Memory: 13.7M
      CGroup: /system.slice/glusterd.service
              └─1903 /usr/sbin/glusterd -p /var/run/glusterd.pid --log-level INFO

May 08 09:23:33 node2.novalocal systemd[1]: /usr/lib/systemd/system/glusterd.service:12: PIDFile= references a path below leg...
May 08 09:23:33 node2.novalocal systemd[1]: Starting GlusterFS, a clustered file-system server...
May 08 09:23:33 node2.novalocal systemd[1]: Started GlusterFS, a clustered file-system server.
May 08 09:23:33 node2.novalocal systemd[1]: /usr/lib/systemd/system/glusterd.service:12: PIDFile= references a path below leg...
lines 1-14/14 (END)

[root@node2 ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        716M   0  716M  0% /dev
tmpfs          732M   0  732M  0% /dev/shm
tmpfs          293M  4.3M 289M  2% /run
tmpfs          4.0M   0  4.0M  0% /sys/fs/cgroup
/dev/mapper/openeuler-root  35G  2.5G  30G  8% /
tmpfs          732M  32K  732M  1% /tmp
/dev/vda1       974M  87M  820M 10% /boot
/dev/vdb1       2.0G  47M  2.0G  3% /exp/vdb1
/dev/vdb2       2.0G  47M  2.0G  3% /exp/vdb2
/dev/vdb3       2.0G  47M  2.0G  3% /exp/vdb3
/dev/vdb5       2.0G  47M  2.0G  3% /exp/vdb5
/dev/vdb6       2.0G  47M  2.0G  3% /exp/vdb6
[root@node2 ~]# 

[root@node3 ~]# systemctl status glusterd
● glusterd.service - GlusterFS, a clustered file-system server
  Loaded: loaded (/usr/lib/systemd/system/glusterd.service; enabled; vendor preset: disabled)
  Active: active (running) since Mon 2023-05-08 09:58:12 CST; 54s ago
    Docs: man:glusterd(8)
    Main PID: 1938 (glusterd)
      Tasks: 24 (limit: 9154)
     Memory: 16.3M
      CGroup: /system.slice/glusterd.service
              └─1938 /usr/sbin/glusterd -p /var/run/glusterd.pid --log-level INFO

May 08 09:58:12 node3.novalocal systemd[1]: /usr/lib/systemd/system/glusterd.service:12: PIDFile= references a path below leg...
May 08 09:58:12 node3.novalocal systemd[1]: Starting GlusterFS, a clustered file-system server...
May 08 09:58:12 node3.novalocal systemd[1]: Started GlusterFS, a clustered file-system server.
May 08 09:58:12 node3.novalocal systemd[1]: /usr/lib/systemd/system/glusterd.service:12: PIDFile= references a path below leg...
lines 1-14/14 (END)

[root@node3 ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        716M   0  716M  0% /dev
tmpfs          732M   0  732M  0% /dev/shm
tmpfs          293M  4.3M 289M  2% /run
tmpfs          4.0M   0  4.0M  0% /sys/fs/cgroup
/dev/mapper/openeuler-root  35G  2.5G  30G  8% /
tmpfs          732M  32K  732M  1% /tmp
/dev/vda1       974M  87M  820M 10% /boot
/dev/vdb1       2.0G  47M  2.0G  3% /exp/vdb1
/dev/vdb2       2.0G  47M  2.0G  3% /exp/vdb2
/dev/vdb3       2.0G  47M  2.0G  3% /exp/vdb3
/dev/vdb5       2.0G  47M  2.0G  3% /exp/vdb5
/dev/vdb6       2.0G  47M  2.0G  3% /exp/vdb6
[root@node3 ~]# 
```

```
[root@node4 ~]# systemctl status glusterd
● glusterd.service - GlusterFS, a clustered file-system server
  Loaded: loaded (/usr/lib/systemd/system/glusterd.service; enabled; vendor preset: disabled)
  Active: active (running) since Mon 2023-05-08 10:02:05 CST; 1min 1s ago
    Docs: man:glusterd(8)
    Main PID: 1945 (glusterd)
       Tasks: 24 (limit: 9154)
      Memory: 13.7M
     CGroup: /system.slice/glusterd.service
             └─1945 /usr/sbin/glusterd -p /var/run/glusterd.pid --log-level INFO

May 08 10:02:05 node4.novalocal systemd[1]: /usr/lib/systemd/system/glusterd.service:12: PIDFile= references a path below leg...
May 08 10:02:05 node4.novalocal systemd[1]: Starting GlusterFS, a clustered file-system server...
May 08 10:02:05 node4.novalocal systemd[1]: Started GlusterFS, a clustered file-system server.
May 08 10:02:05 node4.novalocal systemd[1]: /usr/lib/systemd/system/glusterd.service:12: PIDFile= references a path below leg...
lines 1-14/14 (END)

[root@node4 ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        716M   0  716M  0% /dev
tmpfs          732M   0  732M  0% /dev/shm
tmpfs          293M  4.3M 289M  2% /run
tmpfs          4.0M   0  4.0M  0% /sys/fs/cgroup
/dev/mapper/openeuler-root  35G  2.5G  30G  8% /
tmpfs          732M  32K  732M  1% /tmp
/dev/vda1        974M  87M 820M 10% /boot
/dev/vdb1        2.0G  47M 2.0G  3% /exp/vdb1
/dev/vdb2        2.0G  47M 2.0G  3% /exp/vdb2
/dev/vdb3        2.0G  47M 2.0G  3% /exp/vdb3
/dev/vdb5        2.0G  47M 2.0G  3% /exp/vdb5
/dev/vdb6        2.0G  47M 2.0G  3% /exp/vdb6
[root@node4 ~]# 

[root@node5 ~]# systemctl status glusterd
● glusterd.service - GlusterFS, a clustered file-system server
  Loaded: loaded (/usr/lib/systemd/system/glusterd.service; enabled; vendor preset: disabled)
  Active: active (running) since Mon 2023-05-08 10:05:41 CST; 16s ago
    Docs: man:glusterd(8)
    Main PID: 2000 (glusterd)
       Tasks: 24 (limit: 9154)
      Memory: 13.5M
     CGroup: /system.slice/glusterd.service
             └─2000 /usr/sbin/glusterd -p /var/run/glusterd.pid --log-level INFO

May 08 10:05:41 node5.novalocal systemd[1]: /usr/lib/systemd/system/glusterd.service:12: PIDFile= references a path below leg...
May 08 10:05:41 node5.novalocal systemd[1]: Starting GlusterFS, a clustered file-system server...
May 08 10:05:41 node5.novalocal systemd[1]: Started GlusterFS, a clustered file-system server.
May 08 10:05:41 node5.novalocal systemd[1]: /usr/lib/systemd/system/glusterd.service:12: PIDFile= references a path below leg...
lines 1-14/14 (END)

[root@node5 ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        716M   0  716M  0% /dev
tmpfs          732M   0  732M  0% /dev/shm
tmpfs          293M  4.3M 289M  2% /run
tmpfs          4.0M   0  4.0M  0% /sys/fs/cgroup
/dev/mapper/openeuler-root  35G  2.5G  30G  8% /
tmpfs          732M  32K  732M  1% /tmp
/dev/vda1        974M  87M 820M 10% /boot
/dev/vdb1        2.0G  47M 2.0G  3% /exp/vdb1
/dev/vdb2        2.0G  47M 2.0G  3% /exp/vdb2
/dev/vdb3        2.0G  47M 2.0G  3% /exp/vdb3
/dev/vdb5        2.0G  47M 2.0G  3% /exp/vdb5
/dev/vdb6        2.0G  47M 2.0G  3% /exp/vdb6
[root@node5 ~]# 
```

```
[root@node6 ~]# systemctl status glusterd
● glusterd.service - GlusterFS, a clustered file-system server
   Loaded: loaded (/usr/lib/systemd/system/glusterd.service; enabled; vendor preset: disabled)
     Active: active (running) since Mon 2023-05-08 10:08:13 CST; 11s ago
       Docs: man:glusterd(8)
   Main PID: 1943 (glusterd)
      Tasks: 24 (limit: 9154)
     Memory: 13.5M
        CPU: 0.000 CPU(s) /system.slice/glusterd.service
             └─1943 /usr/sbin/glusterd -p /var/run/glusterd.pid --log-level INFO

May 08 10:08:13 node6.novalocal systemd[1]: /usr/lib/systemd/system/glusterd.service:12: PIDFile= references a path below leg
May 08 10:08:13 node6.novalocal systemd[1]: Starting GlusterFS, a clustered file-system server...
May 08 10:08:13 node6.novalocal systemd[1]: Started GlusterFS, a clustered file-system server.
May 08 10:08:13 node6.novalocal systemd[1]: /usr/lib/systemd/system/glusterd.service:12: PIDFile= references a path below leg
(lines 1-14/14 (END))

[root@node6 ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        716M    0  716M  0% /dev
tmpfs          732M    0  732M  0% /dev/shm
tmpfs          293M  4.3M 289M  2% /run
tmpfs          4.0M    0  4.0M  0% /sys/fs/cgroup
/dev/mapper/openeuler-root  35G  2.5G  30G  8% /
tmpfs          732M  32K  732M  1% /tmp
/dev/vda1        974M  87M  820M 10% /boot
/dev/vdb1        2.0G  47M  2.0G  3% /exp/vdb1
/dev/vdb2        2.0G  47M  2.0G  3% /exp/vdb2
/dev/vdb3        2.0G  47M  2.0G  3% /exp/vdb3
/dev/vdb5        2.0G  47M  2.0G  3% /exp/vdb5
/dev/vdb6        2.0G  47M  2.0G  3% /exp/vdb6
[root@node6 ~]#
```

## Step 7 Configure the GlusterFS trusted storage pool (trusted pool).

On **node1**, add other nodes to the trusted pool.

```
gluster peer probe node2
gluster peer probe node3
gluster peer probe node4
gluster peer probe node5
gluster peer probe node6
```

Check the trusted pool status.

```
gluster peer status
```

```
[root@node1 ~]# gluster peer probe node2
peer probe: success
[root@node1 ~]# gluster peer probe node3
peer probe: success
[root@node1 ~]# gluster peer probe node4
peer probe: success
[root@node1 ~]# gluster peer probe node5
peer probe: success
[root@node1 ~]# gluster peer probe node6
peer probe: success
[root@node1 ~]# gluster peer status
Number of Peers: 5

Hostname: node2
Uuid: 5add33ed-e7cd-46ca-a4cc-bb22fc9b5e3f
State: Peer in Cluster (Connected)

Hostname: node3
Uuid: daa1a255-a9f9-4300-be23-fd46cac522aa
State: Peer in Cluster (Connected)

Hostname: node4
Uuid: 671d2bee-6f7b-49ae-8ecc-e0f6085ce59c
State: Peer in Cluster (Connected)

Hostname: node5
Uuid: 74d88fb8-1b34-4541-8476-d7b1ba2b9440
State: Peer in Cluster (Connected)

Hostname: node6
Uuid: 1a409a6d-152f-4278-be28-f6e06432b63b
State: Peer in Cluster (Connected)
```

Check the list of nodes in the trusted pool.

UUID	Hostname	State
5add33ed-e7cd-46ca-a4cc-bb22fc9b5e3f	node2	Connected
daa1a255-a9f9-4300-be23-fd46cac522aa	node3	Connected
671d2bee-6f7b-49ae-8ecc-e0f6085ce59c	node4	Connected
74d88fb8-1b34-4541-8476-d7b1ba2b9440	node5	Connected
1a409a6d-152f-4278-be28-f6e06432b63b	node6	Connected
695a93ad-8cc5-43d0-8206-38bf3920707e	localhost	Connected

**node1, node2, node3, node4, node5 and node6** have joined the trusted pool. You can also query the trusted pool status and node list on other nodes.

```
[root@node2 ~]# gluster peer status
Number of Peers: 5

Hostname: node1
Uuid: 695a93ad-8cc5-43d0-8206-38bf3920707e
State: Peer in Cluster (Connected)

Hostname: node3
Uuid: daa1a255-a9f9-4300-be23-fd46cac522aa
State: Peer in Cluster (Connected)

Hostname: node4
Uuid: 671d2bee-6f7b-49ae-8ecc-e0f6085ce59c
State: Peer in Cluster (Connected)

Hostname: node5
Uuid: 74d88fb8-1b34-4541-8476-d7b1ba2b9440
State: Peer in Cluster (Connected)

Hostname: node6
Uuid: 1a409a6d-152f-4278-be28-f6e06432b63b
State: Peer in Cluster (Connected)
```

UUID	Hostname	State
695a93ad-8cc5-43d0-8206-38bf3920707e	node1	Connected
daa1a255-a9f9-4300-be23-fd46cac522aa	node3	Connected
671d2bee-6f7b-49ae-8ecc-e0f6085ce59c	node4	Connected
74d88fb8-1b34-4541-8476-d7b1ba2b9440	node5	Connected
1a409a6d-152f-4278-be28-f6e06432b63b	node6	Connected
5add33ed-e7cd-46ca-a4cc-bb22fc9b5e3f	localhost	Connected

### Step 8 Create a test volume.

Use **/dev/vdb1/brick** on **node4** and **/dev/vdb1/brick** on **node5** to create a simple replicated volume **test-volume** for testing client mounting.

```
gluster volume create test-volume replica 2 node4:/exp/vdb1/brick    node5:/exp/vdb1/brick
```

```
[root@node1 ~]# gluster volume create test-volume replica 2 node4:/exp/vdb1/brick    node5:/exp/vdb1/brick
Replica 2 volumes are prone to split-brain. Use Arbiter or Replica 3 to avoid this. See: http://docs.gluster.org/en/latest/Administrator-Guide/Split-brain-and-ways-to-deal-with-it/
Do you still want to continue?
(y/n)
volume create: test-volume: success: please start the volume to access data
[root@node1 ~]#
```

Check volume information.

```
gluster volume info
```

```
[root@node1 ~]# gluster volume info

Volume Name: test-volume
Type: Replicate
Volume ID: d12a17a4-fd43-4dde-ad99-8411499e8645
Status: Created
Snapshot Count: 0
Number of Bricks: 1 x 2 = 2
Transport-type: tcp
Bricks:
Brick1: node4:/exp/vdb1/brick
Brick2: node5:/exp/vdb1/brick
Options Reconfigured:
cluster.granular-entry-heal: on
storage.fips-mode-rchecksum: on
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off
```

The status of **test-volume** is **Created**, and the bricks of the volume are displayed. Start the volume and check the volume information again.

```
gluster volume start test-volume
```

```
[root@node1 ~]# gluster volume start test-volume
volume start: test-volume: success
[root@node1 ~]# gluster volume info

Volume Name: test-volume
Type: Replicate
Volume ID: d12a17a4-fd43-4dde-ad99-8411499e8645
Status: Started
Snapshot Count: 0
Number of Bricks: 1 x 2 = 2
Transport-type: tcp
Bricks:
Brick1: node4:/exp/vdb1/brick
Brick2: node5:/exp/vdb1/brick
Options Reconfigured:
cluster.granular-entry-heal: on
storage.fips-mode-rchecksum: on
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off
```

The volume status changes to **Started**.

#### Step 9    Install the client.

Log in to **Client01** and run the following command to install the client service:

```
yum install -y glusterfs glusterfs-fuse
```

```
[root@Client01 ~]# yum install -y glusterfs glusterfs-fuse
Last metadata expiration check: 1:16:52 ago on Mon Jul 10 10:44:21 UTC 2023.
Dependencies resolved.

=====
          Package           Architecture      Version
=====
Installing:
  glusterfs             x86_64            10.0-8.oe2203
  glusterfs-fuse        x86_64            10.0-8.oe2203

Installing dependencies:
  glusterfs-client-xlators x86_64            10.0-8.oe2203
  gperftools-libs         x86_64            2.9.1-4.oe2203
  libgfrpc0               x86_64            10.0-8.oe2203
  libgfdxdr0              x86_64            10.0-8.oe2203
  libglusterfs0           x86_64            10.0-8.oe2203
  libunwind               x86_64            2:1.5.0-2.oe2203

Transaction Summary
=====
Install  8 Packages

Total download size: 2.1 M
Installed size: 9.5 M
Downloading Packages:
(1/8): gperftools-libs-2.9.1-4.oe2203.x86_64.rpm
(2/8): glusterfs-fuse-10.0-8.oe2203.x86_64.rpm
(3/8): libgfrpc0-10.0-8.oe2203.x86_64.rpm
(4/8): libgfdxdr0-10.0-8.oe2203.x86_64.rpm
(5/8): glusterfs-client-xlators-10.0-8.oe2203.x86_64.rpm
(6/8): glusterfs-10.0-8.oe2203.x86_64.rpm
(7/8): libunwind-1.5.0-2.oe2203.x86_64.rpm
(8/8): libglusterfs0-10.0-8.oe2203.x86_64.rpm

Total
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing :
  Running scriptlet: libunwind-2:1.5.0-2.oe2203.x86_64
  Installing   : libunwind-2:1.5.0-2.oe2203.x86_64
  Installing   : gperftools-libs-2.9.1-4.oe2203.x86_64
  Installing   : libgfrpc0-10.0-8.oe2203.x86_64
  Running scriptlet: libgfrpc0-10.0-8.oe2203.x86_64
  Installing   : libgfdxdr0-10.0-8.oe2203.x86_64
  Running scriptlet: libgfdxdr0-10.0-8.oe2203.x86_64
  Installing   : libglusterfs0-10.0-8.oe2203.x86_64
  Running scriptlet: libglusterfs0-10.0-8.oe2203.x86_64
  Running scriptlet: glusterfs-10.0-8.oe2203.x86_64
  Installing   : glusterfs-10.0-8.oe2203.x86_64
  Running scriptlet: glusterfs-10.0-8.oe2203.x86_64
  Installing   : glusterfs-client-xlators-10.0-8.oe2203.x86_64
  Installing   : glusterfs-fuse-10.0-8.oe2203.x86_64
  Running scriptlet: glusterfs-fuse-10.0-8.oe2203.x86_64
  Verifying    : gperftools-libs-2.9.1-4.oe2203.x86_64
  Verifying    : glusterfs-10.0-8.oe2203.x86_64
  Verifying    : glusterfs-client-xlators-10.0-8.oe2203.x86_64
  Verifying    : glusterfs-fuse-10.0-8.oe2203.x86_64
  Verifying    : libgfrpc0-10.0-8.oe2203.x86_64
  Verifying    : libgfdxdr0-10.0-8.oe2203.x86_64
  Verifying    : libglusterfs0-10.0-8.oe2203.x86_64
  Verifying    : libunwind-2:1.5.0-2.oe2203.x86_64

Installed:
  glusterfs-10.0-8.oe2203.x86_64   glusterfs-client-xlators-10.0-8.oe2203.x86_64   glusterfs-fuse-10.0-8.oe2203.x86_64
  libgfdxdr0-10.0-8.oe2203.x86_64  libglusterfs0-10.0-8.oe2203.x86_64       libunwind-2:1.5.0-2.oe2203.x86_64

Complete!
```

Edit **/etc/hosts** to set host name mapping for each storage node.

```
echo "192.168.1.31 node1" >> /etc/hosts
echo "192.168.1.32 node2" >> /etc/hosts
echo "192.168.1.33 node3" >> /etc/hosts
echo "192.168.1.34 node4" >> /etc/hosts
echo "192.168.1.35 node5" >> /etc/hosts
echo "192.168.1.36 node6" >> /etc/hosts
```

```
[root@Client01 ~]# cat /etc/hosts
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.1.31 node1
192.168.1.32 node2
192.168.1.33 node3
192.168.1.34 node4
192.168.1.35 node5
192.168.1.36 node6
```

## Step 10 Mount the volume to the client.

On **Client01**, create the **/mnt/gfs/test** directory for mounting **test-volume**.

```
mkdir -p /mnt/gfs/test
```

Mount the volume.

```
mount -t glusterfs node1:test-volume /mnt/gfs/test
```

```
[root@Client01 ~]# mount -t glusterfs node1:test-volume /mnt/gfs/test
[root@Client01 ~]# df -h
Filesystem           Size   Used  Avail Use% Mounted on
devtmpfs              1.7G     0  1.7G   0% /dev
tmpfs                 1.7G     0  1.7G   0% /dev/shm
tmpfs                 683M   8.6M  674M   2% /run
tmpfs                 4.0M     0  4.0M   0% /sys/fs/cgroup
/dev/mapper/openeuler-root  35G   2.5G  30G   8% /
tmpfs                 1.7G   32K  1.7G   1% /tmp
/dev/vda1              974M   87M  820M  10% /boot
node1:test-volume      2.0G   68M  2.0G   4% /mnt/gfs/test
```

In **/mnt/gfs/test**, create file **test.txt** and write **This is from Client01** into it.

```
cd /mnt/gfs/test
echo "This is from Client01" > test.txt
```

On **node4** and **node5**, check the content of **/exp/vdb1/brick**.

```
[root@node4 ~]# cd /exp/vdb1/brick
[root@node4 brick]# ls
test.txt
[root@node4 brick]# cat test.txt
This is from Client1
[root@node4 brick]#
```

```
[root@node5 ~]# cd /exp/vdb1/brick
[root@node5 brick]# ls
test.txt
[root@node5 brick]# cat test.txt
This is from Client1
[root@node5 brick]#
```

The same file exists in the bricks on the two nodes, indicating that replicated volume **test-volume** is successfully mounted and has two replicas.

## Step 11 Create and use a distributed volume.

A GlusterFS distributed volume stores created files in different bricks randomly. Create and start distributed volume **gv-dis**.

```
gluster volume create gv-dis node1:/exp/vdb1/brick node2:/exp/vdb1/brick
gluster volume start gv-dis
```

```
[root@node1 ~]# gluster volume create gv-dis node1:/exp/vdb1/brick node2:/exp/vdb1/brick
volume create: gv-dis: success: please start the volume to access data
[root@node1 ~]# gluster volume start gv-dis
volume start: gv-dis: success
[root@node1 ~]# gluster volume info

Volume Name: gv-dis
Type: Distribute
Volume ID: 41d794dd-09d8-4751-8755-3bd295d919ab
Status: Started
Snapshot Count: 0
Number of Bricks: 2
Transport-type: tcp
Bricks:
Brick1: node1:/exp/vdb1/brick
Brick2: node2:/exp/vdb1/brick
Options Reconfigured:
storage.fips-mode-rchecksum: on
transport.address-family: inet
nfs.disable: on

Volume Name: test-volume
Type: Replicate
Volume ID: d12a17a4-fd43-4dde-ad99-8411499e8645
Status: Started
Snapshot Count: 0
Number of Bricks: 1 x 2 = 2
Transport-type: tcp
Bricks:
Brick1: node4:/exp/vdb1/brick
Brick2: node5:/exp/vdb1/brick
Options Reconfigured:
performance.client-io-threads: off
nfs.disable: on
transport.address-family: inet
storage.fips-mode-rchecksum: on
cluster.granular-entry-heal: on
[root@node1 ~]# ]
```

Check the volume list.

```
gluster volume list
```

```
[root@node1 ~]# gluster volume list
gv-dis
test-volume
```

On **Client01**, create mount point **/mnt/gfs/dis** and mount **gv-dis** to it.

```
mkdir -p /mnt/gfs/dis
mount -t glusterfs node1:gv-dis /mnt/gfs/dis
```

```
[root@Client01 ~]# mkdir -p /mnt/gfs/dis
[root@Client01 ~]# mount -t glusterfs node1:gv-dis /mnt/gfs/dis
[root@Client01 ~]# df -h
Filesystem           Size   Used  Avail Use% Mounted on
devtmpfs              1.7G    0  1.7G  0% /dev
tmpfs                 1.7G    0  1.7G  0% /dev/shm
tmpfs                 683M  8.6M  674M  2% /run
tmpfs                 4.0M    0  4.0M  0% /sys/fs/cgroup
/dev/mapper/openeuler-root  35G  2.5G  30G  8% /
tmpfs                 1.7G  32K  1.7G  1% /tmp
/dev/vda1              974M  87M  820M 10% /boot
node1:gv-dis            4.0G 335M  3.7G  9% /mnt/gfs/dis
```

**gv-dis** contains two 2 GB bricks. The size of the mounted **gv-dis** volume is 4 GB, which is the sum of the capacities of the two bricks.

Write five files into **/mnt/gfs/dis**.

```
dd if=/dev/zero of=/mnt/gfs/dis/test1.txt bs=1M count=40
dd if=/dev/zero of=/mnt/gfs/dis/test2.txt bs=1M count=40
dd if=/dev/zero of=/mnt/gfs/dis/test3.txt bs=1M count=40
dd if=/dev/zero of=/mnt/gfs/dis/test4.txt bs=1M count=40
dd if=/dev/zero of=/mnt/gfs/dis/test5.txt bs=1M count=40
```

```
[root@Client01 ~]# ll /mnt/gfs/dis
total 200M
-rw-r--r-- 1 root root 40M May 11:02 test1.txt
-rw-r--r-- 1 root root 40M May 11:02 test2.txt
-rw-r--r-- 1 root root 40M May 11:02 test3.txt
-rw-r--r-- 1 root root 40M May 11:02 test4.txt
-rw-r--r-- 1 root root 40M May 11:02 test5.txt
[root@Client01 ~]#
```

On **node1** and **node2**, check the content of **/exp/vdb1/brick**.

```
[root@node1 brick]# ls
test1.txt  test3.txt  test4.txt
[root@node1 brick]#
```

```
[root@node2 brick]# ls
test2.txt  test5.txt
[root@node2 brick]#
```

The five files are distributed in different bricks, and each file can exist in only one brick.

Step 12 Create and use a replicated volume.

Use **/exp/vdb2/brick** on **node1**, **node2**, and **node3** to create a volume **gv-rep** with three replicas by referring to the networking information and Step 8.

```
gluster volume create gv-rep replic 3 node1:/exp/vdb2/brick node2:/exp/vdb2/brick  
node3:/exp/vdb2/brick  
gluster volume start gv-rep
```

On Client01, mount gv-rep to /mnt/gfs/rep.

```
[root@Client01 ~]# df -h  
Filesystem           Size  Used  Avail Use% Mounted on  
devtmpfs             1.7G   0    1.7G  0% /dev  
tmpfs                1.7G   0    1.7G  0% /dev/shm  
tmpfs                683M  8.6M  674M  2% /run  
tmpfs                4.0M   0    4.0M  0% /sys/fs/cgroup  
/dev/mapper/openeuler-root  35G  2.5G  30G  8% /  
tmpfs                1.7G  32K  1.7G  1% /tmp  
/dev/vda1              974M  87M  820M 10% /boot  
node1:gv-dis            4.0G  535M  3.5G 14% /mnt/gfs/dis  
node1:gv-rep            2.0G  68M  2.0G  4% /mnt/gfs/rep
```

**gv-rep** contains three 2 GB bricks. The size of the mounted **gv-rep** volume is 2 GB, which is a third of the total size.

Write five files into **/mnt/gfs/rep**.

```
dd if=/dev/zero of=/mnt/gfs/rep/test1.txt bs=1M count=40  
dd if=/dev/zero of=/mnt/gfs/rep/test2.txt bs=1M count=40  
dd if=/dev/zero of=/mnt/gfs/rep/test3.txt bs=1M count=40  
dd if=/dev/zero of=/mnt/gfs/rep/test4.txt bs=1M count=40  
dd if=/dev/zero of=/mnt/gfs/rep/test5.txt bs=1M count=40
```

```
[root@Client01 ~]# ll /mnt/gfs/rep  
total 200M  
-rw-r--r-- 1 root root 40M May 11:24 test1.txt  
-rw-r--r-- 1 root root 40M May 11:24 test2.txt  
-rw-r--r-- 1 root root 40M May 11:24 test3.txt  
-rw-r--r-- 1 root root 40M May 11:24 test4.txt  
-rw-r--r-- 1 root root 40M May 11:24 test5.txt  
[root@Client01 ~]#
```

On **node1**, **node2**, and **node3**, check the content of **/exp/vdb2/brick**.

```
[root@node1 brick]# cd /exp/vdb2/brick  
[root@node1 brick]# ll  
total 200M  
-rw-r--r-- 2 root root 40M May 11:24 test1.txt  
-rw-r--r-- 2 root root 40M May 11:24 test2.txt  
-rw-r--r-- 2 root root 40M May 11:24 test3.txt  
-rw-r--r-- 2 root root 40M May 11:24 test4.txt  
-rw-r--r-- 2 root root 40M May 11:24 test5.txt  
[root@node1 brick]#
```

```
[root@node2 brick]# cd /exp/vdb2/brick
[root@node2 brick]# ll
total 200M
-rw-r--r-- 2 root root 40M May 11:24 test1.txt
-rw-r--r-- 2 root root 40M May 11:24 test2.txt
-rw-r--r-- 2 root root 40M May 11:24 test3.txt
-rw-r--r-- 2 root root 40M May 11:24 test4.txt
-rw-r--r-- 2 root root 40M May 11:24 test5.txt
[root@node2 brick]#
```

```
[root@node3 ~]# cd /exp/vdb2/brick
[root@node3 brick]# ll
total 200M
-rw-r--r-- 2 root root 40M May 11:24 test1.txt
-rw-r--r-- 2 root root 40M May 11:24 test2.txt
-rw-r--r-- 2 root root 40M May 11:24 test3.txt
-rw-r--r-- 2 root root 40M May 11:24 test4.txt
-rw-r--r-- 2 root root 40M May 11:24 test5.txt
[root@node3 brick]#
```

In this replicated volume, each brick stores a replica of the files.

### Step 13 Create and use a distributed replicated volume.

Use four bricks to create volume **gv-disrep** based on the networking information.

```
gluster volume create gv-disrep replica 2 node1:/exp/vdb3/brick node2:/exp/vdb3/brick
node3:/exp/vdb3/brick node4:/exp/vdb3/brick
gluster volume start gv-disrep
```

```
[root@node1 brick]# gluster volume info gv-disrep
Volume Name: gv-disrep
Type: Distributed-Replicate
Volume ID: 1fc67f83-54e5-4f9b-ad3b-07e52f22161e
Status: Started
Snapshot Count: 0
Number of Bricks: 2 x 2 = 4
Transport-type: tcp
Bricks:
Brick1: node1:/exp/vdb3/brick
Brick2: node2:/exp/vdb3/brick
Brick3: node3:/exp/vdb3/brick
Brick4: node4:/exp/vdb3/brick
Options Reconfigured:
cluster.granular-entry-heal: on
storage.fips-mode-rchecksum: on
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off
```

The command for creating the distributed replicated volume is similar to that for the replicated volume, differing only in the number of replicas and added bricks.

On Client01, mount gv-disrep to /mnt/gfs/disrep.

```
mkdir -p /mnt/gfs/disrep  
mount -t glusterfs node1:gv-disrep /mnt/gfs/disrep
```

```
[root@Client01 ~]# df -h  
Filesystem Size Used Avail Use% Mounted on  
devtmpfs 1.7G 0 1.7G 0% /dev  
tmpfs 1.7G 0 1.7G 0% /dev/shm  
tmpfs 683M 8.6M 674M 2% /run  
tmpfs 4.0M 0 4.0M 0% /sys/fs/cgroup  
/dev/mapper/openeuler-root 35G 2.5G 30G 8% /  
tmpfs 1.7G 32K 1.7G 1% /tmp  
/dev/vda1 974M 87M 820M 10% /boot  
node1:gv-dis 4.0G 535M 3.5G 14% /mnt/gfs/dis  
node1:gv-rep 2.0G 268M 1.8G 14% /mnt/gfs/rep  
node1:gv-disrep 4.0G 135M 3.9G 4% /mnt/gfs/disrep  
[root@Client01 ~]#
```

**gv-disrep** contains four 2 GB bricks. The size of the mounted **gv-disrep** volume is 4 GB, which is a half of the total capacity. This means that the other half is used as the replica.

Write five files into **/mnt/gfs/disrep**.

```
dd if=/dev/zero of=/mnt/gfs/disrep/test1.txt bs=1M count=40  
dd if=/dev/zero of=/mnt/gfs/disrep/test2.txt bs=1M count=40  
dd if=/dev/zero of=/mnt/gfs/disrep/test3.txt bs=1M count=40  
dd if=/dev/zero of=/mnt/gfs/disrep/test4.txt bs=1M count=40  
dd if=/dev/zero of=/mnt/gfs/disrep/test5.txt bs=1M count=40
```

```
[root@Client01 ~]# ll /mnt/gfs/disrep  
total 200M  
-rw-r--r-- 1 root root 40M May 11:38 test1.txt  
-rw-r--r-- 1 root root 40M May 11:38 test2.txt  
-rw-r--r-- 1 root root 40M May 11:38 test3.txt  
-rw-r--r-- 1 root root 40M May 11:38 test4.txt  
-rw-r--r-- 1 root root 40M May 11:38 test5.txt  
[root@Client01 ~]#
```

On **node1**, **node2**, **node3**, and **node4**, check the content of **/exp/vdb3/brick**.

```
[root@node1 brick]# cd /exp/vdb3/brick  
[root@node1 brick]# ls  
test1.txt test3.txt test4.txt  
[root@node1 brick]#
```

```
[root@node2 brick]# cd /exp/vdb3/brick
[root@node2 brick]# ls
test1.txt  test3.txt  test4.txt
[root@node2 brick]#
```

```
[root@node3 brick]# cd /exp/vdb3/brick
[root@node3 brick]# ls
test2.txt  test5.txt
[root@node3 brick]#
```

```
[root@node4 ~]# cd /exp/vdb3/brick
[root@node4 brick]# ls
test2.txt  test5.txt
[root@node4 brick]#
```

The bricks on **node1** and **node2** are replicas of each other, as are the bricks on **node3** and **node4**. This is because when a distributed replicated volume is created, two adjacent bricks become replicas of each other.

#### Step 14 Create and use a dispersed volume.

Dispersed volumes stripe the encoded data of files, add some redundancy, and store them across multiple bricks in the volume. Use **/exp/vdb5/brick** on **node1**, **node2**, and **node3** to create dispersed volume **gv-disp** by referring to the networking information. Divide each file into three stripes, including two for storage and one for redundancy.

```
gluster volume create gv-disp disperse 3 redundancy 1 node1:/exp/vdb5/brick  node2:/exp/vdb5/brick
node3:/exp/vdb5/brick
gluster volume start gv-disp
```

```
[root@node1 ~]# gluster volume info gv-disp

Volume Name: gv-disp
Type: Disperse
Volume ID: 3357422c-c49e-4d07-acee-eb6e1d4e6743
Status: Started
Snapshot Count: 0
Number of Bricks: 1 x (2 + 1) = 3
Transport-type: tcp
Bricks:
Brick1: node1:/exp/vdb5/brick
Brick2: node2:/exp/vdb5/brick
Brick3: node3:/exp/vdb5/brick
Options Reconfigured:
storage.fips-mode-rchecksum: on
transport.address-family: inet
nfs.disable: on
[root@node1 ~]#
```

On Client01, mount gv-disp to /mnt/gfs/disp.

```
mkdir -p /mnt/gfs/disp
mount -t glusterfs node1:gv-disp /mnt/gfs/disp
```

```
[root@Client01 ~]# df -h
Filesystem           Size  Used Avail Use% Mounted on
devtmpfs              1.7G    0  1.7G  0% /dev
tmpfs                 1.7G    0  1.7G  0% /dev/shm
tmpfs                 683M   17M  666M  3% /run
tmpfs                 4.0M    0  4.0M  0% /sys/fs/cgroup
/dev/mapper/openeuler-root  35G  2.5G  30G  8% /
tmpfs                 1.7G   32K  1.7G  1% /tmp
/dev/vda1              974M   87M  820M 10% /boot
node1:gv-dis            4.0G  535M  3.5G 14% /mnt/gfs/dis
node1:gv-rep             2.0G  268M  1.8G 14% /mnt/gfs/rep
node1:gv-disrep          4.0G  335M  3.7G  9% /mnt/gfs/disrep
node1:gv-disp            4.0G  135M  3.9G  4% /mnt/gfs/disp
```

**gv-disp** contains three 2 GB bricks. The size of the mounted **gv-disp** volume is 4 GB, which means that 2 GB is used for redundancy.

Write five files into **/mnt/gfs/isp**.

```
dd if=/dev/zero of=/mnt/gfs/isp/test1.txt bs=1M count=40
dd if=/dev/zero of=/mnt/gfs/isp/test2.txt bs=1M count=40
dd if=/dev/zero of=/mnt/gfs/isp/test3.txt bs=1M count=40
dd if=/dev/zero of=/mnt/gfs/isp/test4.txt bs=1M count=40
dd if=/dev/zero of=/mnt/gfs/isp/test5.txt bs=1M count=40
```

```
[root@Client01 ~]# ll /mnt/gfs/disp
total 200M
-rw-r--r-- 1 root root 40M May 14:51 test1.txt
-rw-r--r-- 1 root root 40M May 14:51 test2.txt
-rw-r--r-- 1 root root 40M May 14:51 test3.txt
-rw-r--r-- 1 root root 40M May 14:51 test4.txt
-rw-r--r-- 1 root root 40M May 14:51 test5.txt
```

On **node1**, **node2**, and **node3**, check the content of **/exp/vdb5/brick**.

```
[root@node1 ~]# ll /exp/vdb5/brick
total 100M
-rw-r--r-- 2 root root 20M May 14:51 test1.txt
-rw-r--r-- 2 root root 20M May 14:51 test2.txt
-rw-r--r-- 2 root root 20M May 14:51 test3.txt
-rw-r--r-- 2 root root 20M May 14:51 test4.txt
-rw-r--r-- 2 root root 20M May 14:51 test5.txt
[root@node1 ~]#
```

```
[root@node2 ~]# ll /exp/vdb5/brick
total 100M
-rw-r--r-- 2 root root 20M May 14:51 test1.txt
-rw-r--r-- 2 root root 20M May 14:51 test2.txt
-rw-r--r-- 2 root root 20M May 14:51 test3.txt
-rw-r--r-- 2 root root 20M May 14:51 test4.txt
-rw-r--r-- 2 root root 20M May 14:51 test5.txt
[root@node2 ~]#
```

```
[root@node3 ~]# ll /exp/vdb5/brick
total 100M
-rw-r--r-- 2 root root 20M May 14:51 test1.txt
-rw-r--r-- 2 root root 20M May 14:51 test2.txt
-rw-r--r-- 2 root root 20M May 14:51 test3.txt
-rw-r--r-- 2 root root 20M May 14:51 test4.txt
-rw-r--r-- 2 root root 20M May 14:51 test5.txt
[root@node3 ~]#
```

Each TXT file is divided into two data blocks and then encoded into three encode blocks distributed across different bricks.

#### Step 15 Create and use a distributed dispersed volume.

Distributed dispersed and distributed replicated volumes both distribute data across nodes, but differ in the order of operations. Distributed replicated volumes first distribute, then replicate, while distributed dispersed volumes first disperse, then distribute. Use

/exp/vb6/brick on node1, node2, node3, node4, node5, and node6 to create distributed dispersed volume gv-dd. Divide each file into three stripes, including two for storage and one for redundancy.

```
gluster volume create gv-dd disperse 3 redundancy 1 node1:/exp/vdb6/brick node2:/exp/vdb6/brick  
node3:/exp/vdb6/brick node4:/exp/vdb6/brick node5:/exp/vdb6/brick node6:/exp/vdb6/brick  
gluster volume start gv-dd
```

```
[root@node1 ~]# gluster volume info gv-dd  
  
Volume Name: gv-dd  
Type: Distributed-Disperse  
Volume ID: b2f451d3-c1cc-4f6f-b254-1385c955326a  
Status: Started  
Snapshot Count: 0  
Number of Bricks: 2 x (2 + 1) = 6  
Transport-type: tcp  
Bricks:  
Brick1: node1:/exp/vdb6/brick  
Brick2: node2:/exp/vdb6/brick  
Brick3: node3:/exp/vdb6/brick  
Brick4: node4:/exp/vdb6/brick  
Brick5: node5:/exp/vdb6/brick  
Brick6: node6:/exp/vdb6/brick  
Options Reconfigured:  
storage.fips-mode-rchecksum: on  
transport.address-family: inet  
nfs.disable: on  
[root@node1 ~]#
```

On Client01, mount gv-disp to /mnt/gfs/dd.

```
mkdir -p /mnt/gfs/dd  
mount -t glusterfs node1:gv-dd /mnt/gfs/dd
```

```
[root@Client01 ~]# df -h
Filesystem           Size  Used Avail Use% Mounted on
devtmpfs             1.7G   0    1.7G  0% /dev
tmpfs                1.7G   0    1.7G  0% /dev/shm
tmpfs                683M  17M  666M  3% /run
tmpfs                4.0M   0    4.0M  0% /sys/fs/cgroup
/dev/mapper/openeuler-root  35G  2.5G  30G  8% /
tmpfs                1.7G  32K  1.7G  1% /tmp
/dev/vda1              974M  87M  820M 10% /boot
node1:gv-dis          4.0G  535M  3.5G 14% /mnt/gfs/dis
node1:gv-rep          2.0G  268M  1.8G 14% /mnt/gfs/rep
node1:gv-disrep       4.0G  335M  3.7G  9% /mnt/gfs/disrep
node1:gv-disp         4.0G  335M  3.7G  9% /mnt/gfs/disp
node1:gv-dd           8.0G  269M  7.7G  4% /mnt/gfs/dd
[root@Client01 ~]#
```

**gv-disp** contains six 2 GB bricks. The size of the mounted **gv-dd** volume is 8 GB, which means that 4 GB is used for redundancy.

Write five files into **/mnt/gfs/dd**.

```
dd if=/dev/zero of=/mnt/gfs/dd/test1.txt bs=1M count=40
dd if=/dev/zero of=/mnt/gfs/dd/test2.txt bs=1M count=40
dd if=/dev/zero of=/mnt/gfs/dd/test3.txt bs=1M count=40
dd if=/dev/zero of=/mnt/gfs/dd/test4.txt bs=1M count=40
dd if=/dev/zero of=/mnt/gfs/dd/test5.txt bs=1M count=40
```

```
[root@Client01 ~]# ll /mnt/gfs/dd
total 200M
-rw-r--r-- 1 root root 40M May 16:22 test1.txt
-rw-r--r-- 1 root root 40M May 16:22 test2.txt
-rw-r--r-- 1 root root 40M May 16:22 test3.txt
-rw-r--r-- 1 root root 40M May 16:22 test4.txt
-rw-r--r-- 1 root root 40M May 16:22 test5.txt
```

On node1, node2, node3, node4, node5, and node6, check the content of **/exp/vdb6/brick**.

```
[root@node1 mnt]# cd /exp/vdb6/brick
[root@node1 brick]# ll
total 60M
-rw-r--r-- 2 root root 20M May 16:22 test1.txt
-rw-r--r-- 2 root root 20M May 16:22 test3.txt
-rw-r--r-- 2 root root 20M May 16:22 test4.txt
[root@node1 brick]#
```

```
[root@node2 ~]# cd /exp/vdb6/brick
[root@node2 brick]# ll
total 60M
-rw-r--r-- 2 root root 20M May 16:22 test1.txt
-rw-r--r-- 2 root root 20M May 16:22 test3.txt
-rw-r--r-- 2 root root 20M May 16:22 test4.txt
[root@node2 brick]#
```

```
[root@node3 ~]# cd /exp/vdb6/brick
[root@node3 brick]# ll
total 60M
-rw-r--r-- 2 root root 20M May 16:22 test1.txt
-rw-r--r-- 2 root root 20M May 16:22 test3.txt
-rw-r--r-- 2 root root 20M May 16:22 test4.txt
[root@node3 brick]#
```

```
[root@node4 ~]# cd /exp/vdb6/brick
[root@node4 brick]# ll
total 40M
-rw-r--r-- 2 root root 20M May 16:22 test2.txt
-rw-r--r-- 2 root root 20M May 16:22 test5.txt
[root@node4 brick]#
```

```
[root@node5 ~]# cd /exp/vdb6/brick
[root@node5 brick]# ll
total 40M
-rw-r--r-- 2 root root 20M May 16:22 test2.txt
-rw-r--r-- 2 root root 20M May 16:22 test5.txt
[root@node5 brick]#
```

```
[root@node6 ~]# cd /exp/vdb6/brick
[root@node6 brick]# ll
total 40M
-rw-r--r-- 2 root root 20M May 16:22 test2.txt
-rw-r--r-- 2 root root 20M May 16:22 test5.txt
[root@node6 brick]#
```

Files are distributed in two dispersed volumes. One volume consists of **node1**, **node2**, and **node3**, the other of **node4**, **node5**, and **node6**. In each dispersed volume, a file is divided into two blocks, encoded into three blocks using erasure coding, and then stored in different bricks.

## Step 16 Add a brick to a volume

Adds a brick to a distributed volume. Run the following command to add **/exp/vdb1/brick** in **node3** to the **gv-dis** volume:

```
gluster volume add-brick gv-dis node3:/exp/vdb1/brick
```

```
[root@node1 brick]# gluster volume add-brick gv-dis node3:/exp/vdb1/brick
volume add-brick: success
[root@node1 brick]# gluster volume info gv-dis

Volume Name: gv-dis
Type: Distribute
Volume ID: 41d794dd-09d8-4751-8755-3bd295d919ab
Status: Started
Snapshot Count: 0
Number of Bricks: 3
Transport-type: tcp
Bricks:
Brick1: node1:/exp/vdb1/brick
Brick2: node2:/exp/vdb1/brick
Brick3: node3:/exp/vdb1/brick
Options Reconfigured:
storage.fips-mode-rchecksum: on
transport.address-family: inet
nfs.disable: on
[root@node1 brick]#
```

The brick has been added. On **Client01**, check the volume capacity.

```
[root@Client01 ~]# df -h
Filesystem           Size  Used  Avail Use% Mounted on
devtmpfs              1.7G   0    1.7G  0% /dev
tmpfs                 1.7G   0    1.7G  0% /dev/shm
tmpfs                 683M  17M   666M  3% /run
tmpfs                 4.0M   0    4.0M  0% /sys/fs/cgroup
/dev/mapper/openeuler-root  35G  2.5G   30G  8% /
tmpfs                 1.7G  32K   1.7G  1% /tmp
/dev/vda1              974M  87M   820M 10% /boot
node1:gv-dis            6.0G  802M   5.2G 14% /mnt/gfs/dis
node1:gv-rep             2.0G  268M   1.8G 14% /mnt/gfs/rep
node1:gv-disrep          4.0G  335M   3.7G  9% /mnt/gfs/disrep
node1:gv-disp             4.0G  335M   3.7G  9% /mnt/gfs/disp
node1:gv-dd              8.0G  469M   7.6G  6% /mnt/gfs/dd
```

Create three files in **/mnt/gfs/dis**.

```
dd if=/dev/zero of=/mnt/gfs/dis/add1.txt bs=1M count=40
dd if=/dev/zero of=/mnt/gfs/dis/add2.txt bs=1M count=40
dd if=/dev/zero of=/mnt/gfs/dis/add3.txt bs=1M count=40
```

```
[root@Client01 ~]# ll /mnt/gfs/dis
total 320M
-rw-r--r-- 1 root root 40M May 16:36 add1.txt
-rw-r--r-- 1 root root 40M May 16:36 add2.txt
-rw-r--r-- 1 root root 40M May 16:36 add3.txt
-rw-r--r-- 1 root root 40M May 11:02 test1.txt
-rw-r--r-- 1 root root 40M May 11:02 test2.txt
-rw-r--r-- 1 root root 40M May 11:02 test3.txt
-rw-r--r-- 1 root root 40M May 11:02 test4.txt
-rw-r--r-- 1 root root 40M May 11:02 test5.txt
[root@Client01 ~]#
```

On node1, node2, and node3, check the files in /exp/vdb1/brick.

```
[root@node1 brick]# ll /exp/vdb1/brick
total 160M
-rw-r--r-- 2 root root 40M May 16:36 add1.txt
-rw-r--r-- 2 root root 40M May 11:02 test1.txt
-rw-r--r-- 2 root root 40M May 11:02 test3.txt
-rw-r--r-- 2 root root 40M May 11:02 test4.txt
[root@node1 brick]#
```

```
[root@node2 brick]# ll /exp/vdb1/brick
total 160M
-rw-r--r-- 2 root root 40M May 16:36 add2.txt
-rw-r--r-- 2 root root 40M May 16:36 add3.txt
-rw-r--r-- 2 root root 40M May 11:02 test2.txt
-rw-r--r-- 2 root root 40M May 11:02 test5.txt
```

```
[root@node3 brick]# ll /exp/vdb1/brick
total 0
[root@node3 brick]#
```

- Question: Why are the files not stored in the newly added brick?

Answer: A brick added to a GlusterFS volume can be used only after the volume is rebalanced.

Rebalance the volume.

```
gluster volume rebalance gv-dis start
gluster volume rebalance gv-dis status
```

```
[root@node1 brick]# gluster volume rebalance gv-dis start
volume rebalance: gv-dis: success: Rebalance on gv-dis has been started successfully. Use rebalance status command to check status of the rebalance process.
ID: 1dbc48ec-6414-4e8d-8758-82a60b31f5ec
[root@node1 brick]# gluster volume rebalance gv-dis status
      Node  Rebalanced-files    size   scanned   failures   skipped
status  run time in h:m:s
-----  -----  -----  -----  -----  -----
completed  0:00:00      node2       0     0Bytes      4        0        0      c
completed  0:00:00      node3       0     0Bytes      0        0        0      c
completed  0:00:00      localhost   1    40.0MB      4        0        0      c
volume rebalance: gv-dis: success
[root@node1 brick]#
```

On **node1**, **node2**, and **node3**, check the files in **/exp/vdb1/brick** again.

```
[root@node1 brick]# ll /exp/vdb1/brick
total 120M
-rw-r--r-- 2 root root 40M May  9 16:36 add1.txt
-rw-r--r-- 2 root root 40M May  9 11:02 test1.txt
-rw-r--r-- 2 root root 40M May  9 11:02 test3.txt
[root@node1 brick]#
```

```
[root@node2 brick]# ll /exp/vdb1/brick
total 160M
-rw-r--r-- 2 root root 40M May  9 16:36 add2.txt
-rw-r--r-- 2 root root 40M May  9 16:36 add3.txt
-rw-r--r-- 2 root root 40M May  9 11:02 test2.txt
-rw-r--r-- 2 root root 40M May  9 11:02 test5.txt
[root@node2 brick]#
```

```
[root@node3 brick]# ll /exp/vdb1/brick
total 40M
-rw-r--r-- 2 root root 40M May  9 11:02 test4.txt
[root@node3 brick]#
```

A file exists in the brick on **node3**.

#### Step 17 Remove a brick from a volume.

Run the following command to remove **/exp/vdb1/brick** on **node1** from **gv-dis**:

```
gluster volume remove-brick gv-dis node1:/exp/vdb1/brick start
gluster volume remove-brick gv-dis node1:/exp/vdb1/brick status
gluster volume remove-brick gv-dis node1:/exp/vdb1/brick commit
```

```
[root@node1 brick]# gluster volume remove-brick gv-dis node1:/exp/vdb1/brick
Usage:
volume remove-brick <VOLNAME> [<replica <COUNT>>] <BRICK> ... <start|stop|status|commit|force>
[root@node1 brick]# gluster volume remove-brick gv-dis node1:/exp/vdb1/brick start
It is recommended that remove-brick be run with cluster.force-migration option disabled to prevent possible data corruption. Doing so will ensure that files that receive writes during migration will not be migrated and will need to be manually copied after the remove-brick commit operation. Please check the value of the option and update accordingly.
Do you want to continue with your current cluster.force-migration settings? (y/n) y
volume remove-brick start: success
ID: 22e97396-76b9-4378-acc4-c9a20a72673a
[root@node1 brick]# gluster volume remove-brick gv-dis node1:/exp/vdb1/brick status
          Node Rebalanced-files      size    scanned   failures   skipped
          status  run time in h:m:s
          -----
          localhost           3     120.0MB        3          0          0
          completed       0:00:01
[root@node1 brick]# gluster volume remove-brick gv-dis node1:/exp/vdb1/brick commit
volume remove-brick commit: success
Check the removed bricks to ensure all files are migrated.
If files with data are found on the brick path, copy them via a gluster mount point before re-purposing the removed brick.
[root@node1 brick]#
```

Check /exp/vdb1/brick on node1, node2, and node3 again.

```
[root@node1 brick]# ll /exp/vdb1/brick
total 0
```

```
[root@node2 brick]# ll /exp/vdb1/brick
total 280M
-rw-r--r-- 2 root root 40M May  9 16:36 add1.txt
-rw-r--r-- 2 root root 40M May  9 16:36 add2.txt
-rw-r--r-- 2 root root 40M May  9 16:36 add3.txt
-rw-r--r-- 2 root root 40M May  9 11:02 test1.txt
-rw-r--r-- 2 root root 40M May  9 11:02 test2.txt
-rw-r--r-- 2 root root 40M May  9 11:02 test3.txt
-rw-r--r-- 2 root root 40M May  9 11:02 test5.txt
```

```
[root@node3 brick]# ll /exp/vdb1/brick
total 40M
-rw-r--r-- 2 root root 40M May  9 11:02 test4.txt
[root@node3 brick]#
```

No file exists in /exp/vdb1/brick on node1.

Step 18 Delete a volume.

Check the current volume list.

```
gluster volume list
```

```
[root@node1 brick]# gluster volume list
gv-dd
gv-dis
gv-disp
gv-disrep
gv-rep
test-volume
```

Check the status of **test-volume**.

```
gluster volume status test-volume
```

```
[root@node1 ~]# gluster volume status test-volume
Status of volume: test-volume
Gluster process
-----
Brick node4:/exp/vdb1/brick          57837   0      Y    1798
Brick node5:/exp/vdb1/brick          54794   0      Y    1505
Self-heal Daemon on localhost        N/A     N/A    Y    6537
Self-heal Daemon on node6           N/A     N/A    Y    1539
Self-heal Daemon on node2           N/A     N/A    Y    1737
Self-heal Daemon on node3           N/A     N/A    Y    1601
Self-heal Daemon on node5           N/A     N/A    Y    1566
Self-heal Daemon on node4           N/A     N/A    Y    1853

Task Status of Volume test-volume
-----
There are no active volume tasks

[root@node1 ~]#
```

Stop **test-volume**.

```
gluster volume stop test-volume
```

```
[root@node1 ~]# gluster volume stop test-volume
Stopping volume will make its data inaccessible. Do you want to continue? (y/n) y
volume stop: test-volume: success
[root@node1 ~]#
```

Delete **test-volume**.

```
gluster volume delete test-volume
```

```
[root@node1 ~]# gluster volume delete test-volume
Deleting volume will erase all information about the volume. Do you want to continue? (y/n) y
volume delete: test-volume: success
[root@node1 ~]#
```

Check the volume list again. **test-volume** has been deleted.

```
gluster volume list
```

```
[root@node1 ~]# gluster volume list
gv-dd
gv-dis
gv-disp
gv-disrep
gv-rep
[root@node1 ~]#
```

### Step 19 Perform a disaster resilience test.

Stop the glusterd service on **node1**.

```
[root@node1 brick]# systemctl stop glusterd
[root@node1 brick]# systemctl status glusterd
● glusterd.service - GlusterFS, a clustered file-system server
   Loaded: loaded (/usr/lib/systemd/system/glusterd.service; enabled; vendor preset: disabled)
   Active: inactive (dead) since Tue 2023-05-09 17:01:49 CST; 1min 12s ago
     Docs: man:glusterd(8)
  Process: 935 ExecStart=/usr/sbin/glusterd -p /var/run/glusterd.pid --log-level $LOG_LEVEL ${GLUSTER
 Main PID: 948 (code=exited, status=15)
    Tasks: 173 (limit: 9154)
   Memory: 537.1M
      CGroup: /system.slice/glusterd.service
              ├─1522 /usr/sbin/glusterfs -s localhost --volfile-id shd/test-volume -p /var/run/gluster/
              ├─2356 /usr/sbin/glusterfsd -s node1 --volfile-id gv-rep.node1.exp-vdb2-brick -p /var/run/
              ├─2465 /usr/sbin/glusterfsd -s node1 --volfile-id gv-disrep.node1.exp-vdb3-brick -p /var/
              ├─4719 /usr/sbin/glusterfsd -s node1 --volfile-id gv-disp.node1.exp-vdb5-brick -p /var/run/
              └─5467 /usr/sbin/glusterfsd -s node1 --volfile-id gv-dd.node1.exp-vdb6-brick -p /var/run/

May 09 08:52:03 node1.novalocal systemd[1]: Starting GlusterFS, a clustered file-system server...
May 09 08:52:03 node1.novalocal systemd[1]: Started GlusterFS, a clustered file-system server.
May 09 17:01:49 node1.novalocal systemd[1]: Stopping GlusterFS, a clustered file-system server...
May 09 17:01:49 node1.novalocal systemd[1]: glusterd.service: Deactivated successfully.
May 09 17:01:49 node1.novalocal systemd[1]: glusterd.service: Unit process 1522 (glusterfs) remains running
May 09 17:01:49 node1.novalocal systemd[1]: glusterd.service: Unit process 2356 (glusterfsd) remains running
May 09 17:01:49 node1.novalocal systemd[1]: glusterd.service: Unit process 2465 (glusterfsd) remains running
May 09 17:01:49 node1.novalocal systemd[1]: glusterd.service: Unit process 4719 (glusterfsd) remains running
May 09 17:01:49 node1.novalocal systemd[1]: glusterd.service: Unit process 5467 (glusterfsd) remains running
May 09 17:01:49 node1.novalocal systemd[1]: Stopped GlusterFS, a clustered file-system server.
```

On another node, check the list of nodes in the trusted pool.

```
[root@node2 brick]# gluster pool list
          UUID                Hostname        State
1a409a6d-152f-4278-be28-f6e06432b63b    node6        Connected
671d2bee-6f7b-49ae-8ecc-e0f6085ce59c    node4        Connected
74d88fb8-1b34-4541-8476-d7b1ba2b9440    node5        Connected
695a93ad-8cc5-43d0-8206-38bf3920707e    node1        Disconnected
daa1a255-a9f9-4300-be23-fd46cac522aa    node3        Connected
5add33ed-e7cd-46ca-a4cc-bb22fc9b5e3f    localhost    Connected
[root@node2 brick]#
```

**node1** is disconnected. Check the status of different types of volumes.

```
[root@node2 brick]# gluster volume status gv-rep
Status of volume: gv-rep
Gluster process
-----
Brick node2:/exp/vdb2/brick          TCP Port  RDMA Port  Online  Pid
Brick node3:/exp/vdb2/brick          59044     0           Y        5782
Brick node3:/exp/vdb2/brick          50354     0           Y        5514
Self-heal Daemon on localhost        N/A       N/A         Y        1737
Self-heal Daemon on node6            N/A       N/A         Y        1539
Self-heal Daemon on node4            N/A       N/A         Y        1853
Self-heal Daemon on node3            N/A       N/A         Y        1601
Self-heal Daemon on node5            N/A       N/A         Y        1566

Task Status of Volume gv-rep
-----
There are no active volume tasks
```

```
[root@node2 brick]# gluster volume status gv-disrep
Status of volume: gv-disrep
Gluster process
-----
Brick node2:/exp/vdb3/brick          TCP Port  RDMA Port  Online  Pid
Brick node3:/exp/vdb3/brick          52388     0           Y        5884
Brick node4:/exp/vdb3/brick          53638     0           Y        5663
Brick node4:/exp/vdb3/brick          51148     0           Y        5900
Self-heal Daemon on localhost        N/A       N/A         Y        1737
Self-heal Daemon on node6            N/A       N/A         Y        1539
Self-heal Daemon on node4            N/A       N/A         Y        1853
Self-heal Daemon on node3            N/A       N/A         Y        1601
Self-heal Daemon on node5            N/A       N/A         Y        1566

Task Status of Volume gv-disrep
-----
There are no active volume tasks
```

```
[root@node2 brick]# gluster volume status gv-disp
Status of volume: gv-disp
Gluster process
-----
Brick node2:/exp/vdb5/brick          TCP Port  RDMA Port  Online  Pid
Brick node3:/exp/vdb5/brick          53427     0           Y        8114
Brick node3:/exp/vdb5/brick          59015     0           Y        7143
Self-heal Daemon on localhost        N/A       N/A         Y        1737
Self-heal Daemon on node6            N/A       N/A         Y        1539
Self-heal Daemon on node4            N/A       N/A         Y        1853
Self-heal Daemon on node5            N/A       N/A         Y        1566
Self-heal Daemon on node3            N/A       N/A         Y        1601

Task Status of Volume gv-disp
-----
There are no active volume tasks
```

```
[root@node2 brick]# gluster volume status gv-dd
Status of volume: gv-dd
Gluster process
-----  
Brick node2:/exp/vdb6/brick          58433    0      Y    9014  
Brick node3:/exp/vdb6/brick          50218    0      Y    7765  
Brick node4:/exp/vdb6/brick          50206    0      Y    7857  
Brick node5:/exp/vdb6/brick          50614    0      Y    7223  
Brick node6:/exp/vdb6/brick          59713    0      Y    6053  
Self-heal Daemon on localhost        N/A      N/A    Y    1737  
Self-heal Daemon on node6           N/A      N/A    Y    1539  
Self-heal Daemon on node4           N/A      N/A    Y    1853  
Self-heal Daemon on node5           N/A      N/A    Y    1566  
Self-heal Daemon on node3           N/A      N/A    Y    1601  
  
Task Status of Volume gv-dd
-----  
There are no active volume tasks
```

No volume contains bricks in **node1**. On **Client01**, check the mounting status of different volumes.

```
[root@Client01 dis]# df -h
Filesystem            Size  Used Avail Use% Mounted on
devtmpfs              1.7G   0    1.7G  0% /dev
tmpfs                 1.7G   0    1.7G  0% /dev/shm
tmpfs                 683M  17M  666M  3% /run
tmpfs                 4.0M   0    4.0M  0% /sys/fs/cgroup
/dev/mapper/openeuler-root  35G  2.5G  30G  8% /
tmpfs                 1.7G  32K  1.7G  1% /tmp
/dev/vda1              974M  87M  820M 10% /boot
node1:gv-dis           4.0G  535M  3.5G 14% /mnt/gfs/dis
node1:gv-rep           2.0G  268M  1.8G 14% /mnt/gfs/rep
node1:gv-disrep        4.0G  335M  3.7G  9% /mnt/gfs/disrep
node1:gv-disp          4.0G  335M  3.7G  9% /mnt/gfs/disp
node1:gv-dd            8.0G  469M  7.6G  6% /mnt/gfs/dd
[root@Client01 dis]#
```

All volumes are mounted normally. Check files in different volumes.

```
[root@Client01 dis]# ll /mnt/gfs/dis
total 320M
-rw-r--r-- 1 root root 40M May  9 16:36 add1.txt
-rw-r--r-- 1 root root 40M May  9 16:36 add2.txt
-rw-r--r-- 1 root root 40M May  9 16:36 add3.txt
-rw-r--r-- 1 root root 40M May  9 11:02 test1.txt
-rw-r--r-- 1 root root 40M May  9 11:02 test2.txt
-rw-r--r-- 1 root root 40M May  9 11:02 test3.txt
-rw-r--r-- 1 root root 40M May  9 11:02 test4.txt
-rw-r--r-- 1 root root 40M May  9 11:02 test5.txt
```

```
[root@Client01 dis]# ll /mnt/gfs/rep
total 200M
-rw-r--r-- 1 root root 40M May  9 11:24 test1.txt
-rw-r--r-- 1 root root 40M May  9 11:24 test2.txt
-rw-r--r-- 1 root root 40M May  9 11:24 test3.txt
-rw-r--r-- 1 root root 40M May  9 11:24 test4.txt
-rw-r--r-- 1 root root 40M May  9 11:24 test5.txt
```

```
[root@Client01 dis]# ll /mnt/gfs/disrep
total 200M
-rw-r--r-- 1 root root 40M May  9 11:38 test1.txt
-rw-r--r-- 1 root root 40M May  9 11:38 test2.txt
-rw-r--r-- 1 root root 40M May  9 11:38 test3.txt
-rw-r--r-- 1 root root 40M May  9 11:38 test4.txt
-rw-r--r-- 1 root root 40M May  9 11:38 test5.txt
```

```
[root@Client01 dis]# ll /mnt/gfs/disp
total 201M
-rw-r--r-- 1 root root 40M May  9 15:11 test1.txt
-rw-r--r-- 1 root root 40M May  9 15:11 test2.txt
-rw-r--r-- 1 root root 40M May  9 15:11 test3.txt
-rw-r--r-- 1 root root 40M May  9 15:11 test4.txt
-rw-r--r-- 1 root root 40M May  9 15:23 test5.txt
-rw-r--r-- 1 root root 512 May  9 15:25 test6.txt
```

```
[root@Client01 dis]# ll /mnt/gfs/dd
total 200M
-rw-r--r-- 1 root root 40M May  9 16:22 test1.txt
-rw-r--r-- 1 root root 40M May  9 16:22 test2.txt
-rw-r--r-- 1 root root 40M May  9 16:22 test3.txt
-rw-r--r-- 1 root root 40M May  9 16:22 test4.txt
-rw-r--r-- 1 root root 40M May  9 16:22 test5.txt
```

Files in all volumes exist.

Huawei openEuler Certification Training

# HCIP-openEuler

## Lab Guide

ISSUE: 1.0



HUAWEI TECHNOLOGIES CO., LTD

**Copyright © Huawei Technologies Co., Ltd. 2024. All rights reserved.**

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

### **Trademarks and Permissions**



HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

### **Notice**

The purchased products, services, and features are stipulated by the commercial contract made between Huawei and the customer. All or partial products, services, and features described in this document may not be within the purchased scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

## **Huawei Technologies Co., Ltd.**

Address: Huawei Industrial Base Bantian, Longgang Shenzhen 518129  
People's Republic of China

Website: <https://e.huawei.com>

## Huawei Certification System

Huawei Certification is an integral part of the company's Platform + Ecosystem strategy. It supports the development of ICT infrastructure that features Cloud-Pipe-Device synergy. Our certification is always evolving to reflect the latest trends in ICT development. Huawei Certification consists of three categories: ICT Infrastructure Certification, Basic Software & Hardware Certification, and Cloud Platform & Services Certification, making it the most extensive technical certification program in the industry.

Huawei offers three levels of certification: Huawei Certified ICT Associate (HCIA), Huawei Certified ICT Professional (HCIP), and Huawei Certified ICT Expert (HCIE).

Our programs cover all ICT fields and follow the industry's trend of ICT convergence. With our leading talent development system and certification standards, we are committed to fostering new digital ICT talent and building a sound ICT talent ecosystem.

HCIP-openEuler is mainly for frontline engineers from Huawei and representative offices and readers who wish to learn openEuler operations and maintenance (O&M) technologies. HCIP-openEuler certification covers common openEuler enterprise service management, openEuler HA cluster architecture, openEuler storage management, openEuler automated O&M, Linux shell scripts, openEuler system security hardening, and openEuler system monitoring.

Huawei certification helps you unlock opportunities to advance your career and take one more step towards the top of the industry.

## Huawei Career Certification

Cloud Platform & Services Domain		Cloud Computing	Cloud Service	Big Data	GaussDB	IoT
Basic Software & Hardware Domain		openEuler	openGauss	HarmonyOS	AI	Kunpeng
IT Infrastructure Domain		Storage	Collaboration	Intelligent Vision	Digital Power	MDC
Datacom Domain		Datacom		WLAN		Security
Optical Domain		Transmission			Access	
Wireless Domain		5G			LTE	
Huawei Certified ICT Expert						
Huawei Certified ICT Professional						
Huawei Certified ICT Associate						

# About This Document

---

## Overview

This document is an HCIP-openEuler certification training course and is intended for trainees who are going to take the HCIP-openEuler exam or readers who want to learn how to build enterprise services, shell scripts, or perform automated O&M using Zabbix or SaltStack on openEuler and other Linux distributions.

## Description

This lab guide consists of two labs, one for Ansible and the other for SaltStack automated O&M.

- Lab 1 is related to Ansible. This lab introduces basic Ansible operations, practices common function modules, and simply compiles and executes the playbook.
- Lab 2 is related to SaltStack. This lab introduces the remote execution function and configuration management function of SaltStack.

## Background Knowledge Required

This course is for Huawei's basic certification. To better understand this course, the intended audiences are advised to meet the following requirement:

- Have basic Linux knowledge. You are advised to complete HCIA-openEuler learning and pass the HCIA-openEuler certification exam.

## Lab Environment Overview

Four virtual machines (VMs) are used in this lab. One is the controller of the automation tool, and the other three are controlled hosts. The following table lists the IP addresses of the VMs.

Role	Host Name	Specification	IP Address
Controller	Ansible or SaltStack	2 vCPUs   4 GiB   s7.large.2	192.168.1.60
Controlled host 1	Zabbix	1 vCPU   1 GiB   s7.small.1	192.168.1.4
Controlled host 2	Nginx1	1 vCPU   1 GiB   s7.small.1	192.168.1.14
Controlled host 3	Nginx2	1 vCPU   1 GiB   s7.small.1	192.168.1.15

# Contents

---

<b>About This Document .....</b>	<b>3</b>
Overview .....	3
Description .....	3
Background Knowledge Required .....	3
Lab Environment Overview.....	3
<b>1 Basic Operations of Ansible .....</b>	<b>1</b>
1.1 Installing and Configuring the Ansible Controller.....	1
1.2 Basic Operations of the Ansible Command .....	2
1.3 Practice of Common Ansible Modules.....	6
1.3.1 Practice of the command Module .....	6
1.3.2 Practice of the shell Module .....	6
1.3.3 Practice of the script Module .....	7
1.3.4 Practice of the copy Module.....	7
1.3.5 Practice of the fetch Module .....	8
1.3.6 Practice of the file Module.....	9
1.3.7 Practice of the archive and unarchive Modules .....	11
1.4 Playbook Comprehensive Practice .....	12
1.4.1 Environment Preparation.....	12
1.4.2 Lab Practice.....	12
<b>2 Basic Operations of SaltStack.....</b>	<b>15</b>
2.1 Installing and Configuring SaltStack.....	15
2.1.1 Installing salt-master and salt-minion.....	15
2.1.2 Configuring the Master and Minion Authentication .....	17
2.1.3 Adding a Client in SSH Mode .....	17
2.2 SaltStack Remote Execution Function Practice .....	18
2.2.1 Specifying the Target.....	18
2.2.2 Practice of Remote Execution Function Modules – cmd.....	19
2.2.3 Practice of Remote Execution Function Modules – pkg .....	20
2.2.4 Practice of Remote Execution Function Modules – service.....	21
2.2.5 Practice of Remote Execution Function Modules – network .....	22
2.2.6 Practice of Remote Execution Function Modules – file .....	23
2.3 SaltStack Configuration Management Function Practice.....	25
2.3.1 Configuration Management Script Practice .....	25
2.3.2 Composing the top.sls File.....	27

# 1

# Basic Operations of Ansible

---

## 1.1 Installing and Configuring the Ansible Controller

### Step 1 Install Ansible.

Log in to the Ansible host and run the following command to install Ansible:

```
yum install -y ansible
```

After the installation is complete, run the following command to check Ansible:

```
ansible --version
```

```
[root@Ansible ~]# ansible --version
ansible 2.9.27
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/root/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3.9/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.9.9 (main, Mar 15 2022, 00:00:00) [GCC 10.3.1]
```

### Step 2 Configure Ansible.

Modify the Ansible configuration file and uncomment the **host\_key\_checking** parameter, as shown in the following figure:

```
#plugins_path      = /etc/ansible/
# uncomment this to disable SSH host_key_checking = False
# change the default callback,
```

Modify the Ansible host list, add IP addresses of hosts Zabbix, Nginx1, and Nginx2 to the list, and group the hosts as shown in the following figure:

```
[Nginx]
192.168.1.14
192.168.1.15
[Zabbix]
192.168.1.4
```

### Step 3 Configure password-free login for the controlled hosts.

Run the following commands to configure SSH password-free login from the controller to the controlled hosts:

```
ssh-keygen  
ssh-copy-id 192.168.1.4  
ssh-copy-id 192.168.1.14  
ssh-copy-id 192.168.1.15
```

## 1.2 Basic Operations of the Ansible Command

In this lab, the **ping** module is used to practice the basic operations of the Ansible command.

### Step 1 Ansible uses the **ping** module to check whether the communication with the controlled hosts is normal.

Run the following command to invoke the **ping** module and check whether the controller can communicate with the controlled hosts:

```
ansible all -m ping
```

```
[root@Ansible ~]# ansible all -m ping  
[WARNING]: Platform linux on host 192.168.1.4 is using the d  
https://docs.ansible.com/ansible/2.9/reference_appendices/in  
192.168.1.4 | SUCCESS => {  
    "ansible_facts": {  
        "discovered_interpreter_python": "/usr/bin/python3"  
    },  
    "changed": false,  
    "ping": "pong"  
}  
[WARNING]: Platform linux on host 192.168.1.14 is using the d  
https://docs.ansible.com/ansible/2.9/reference_appendices/in  
192.168.1.14 | SUCCESS => {  
    "ansible_facts": {  
        "discovered_interpreter_python": "/usr/bin/python3"  
    },  
    "changed": false,  
    "ping": "pong"  
}  
[WARNING]: Platform linux on host 192.168.1.15 is using the d  
https://docs.ansible.com/ansible/2.9/reference_appendices/in  
192.168.1.15 | SUCCESS => {  
    "ansible_facts": {  
        "discovered_interpreter_python": "/usr/bin/python3"  
    },  
    "changed": false,  
    "ping": "pong"  
}
```

The command output contains warning information. You can add **interpreter\_python = auto\_legacy\_silent** to [default] in the configuration file to clear the warning.

## Step 2 Practice the Ansible -k option.

Use the **-k** option to enter the SSH password in interactive mode. The method is as follows:

```
ansible all -k -m ping
```

```
[root@Ansible ~]# ansible all -k -m ping
SSH password:
192.168.1.15 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

## Step 3 Use the host tag to execute tasks.

**all** indicates all hosts in the host list. Use the host tag to specify the host that runs the command. The command is as follows:

```
ansible Nginx -m ping
```

```
[root@Ansible ~]# ansible Nginx -m ping
192.168.1.15 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
192.168.1.14 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

- Task: Check the communication status between the host in the Zabbix group and the controller.

```
[root@Ansible ~]# ansible Zabbix -m ping
192.168.1.4 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

Step 4 Use **--list-host** to list the corresponding hosts.

**--list-host** can be abbreviated as **--list**, as shown in the following figure:

```
[root@Ansible ~]# ansible Nginx --list
hosts (2):
  192.168.1.14
  192.168.1.15
```

Step 5 Logical relationship of Ansible.

Run the following command to check the communication between the Zabbix or Nginx host group and the controller.

```
ansible "Zabbix:Nginx" -m ping
```

```
[root@Ansible ~]# ansible "Zabbix:Nginx" -m ping
192.168.1.4 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
192.168.1.15 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
192.168.1.14 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

Run the following command to check the communication between the shared hosts in the Zabbix and Nginx host groups and the controller:

```
ansible "Zabbix:&Nginx" -m ping
```

```
[root@Ansible ~]# ansible "Zabbix:&Nginx" -m ping  
[WARNING]: No hosts matched, nothing to do
```

- Question: Why are there no matched hosts?

Answer: The preceding information is displayed because the two host groups do not have shared hosts.

Run the following command to check the communication between the controller and the hosts that do not belong to the Nginx host group:

```
ansible ':!Nginx' -m ping
```

```
[root@Ansible ~]# ansible ':!Nginx' -m ping  
192.168.1.4 | SUCCESS => {  
    "ansible_facts": {  
        "discovered_interpreter_python": "/usr/bin/python3"  
    },  
    "changed": false,  
    "ping": "pong"  
}
```

## Step 6 Check the Ansible execution process.

Use the **-v** parameter to check the Ansible command execution process, as shown in the following figure:

```
ansible ':!Nginx' -v -m ping
```

```
[root@Ansible ~]# ansible ':!Nginx' -v -m ping  
Using /etc/ansible/ansible.cfg as config file  
192.168.1.4 | SUCCESS => {  
    "ansible_facts": {  
        "discovered_interpreter_python": "/usr/bin/python3"  
    },  
    "changed": false,  
    "ping": "pong"  
}
```

- Task: Run **-vv** and **-vvv** to check more detailed execution process.

## 1.3 Practice of Common Ansible Modules

### 1.3.1 Practice of the command Module

Requirement: Copy **/etc/passwd** from the Nginx host group to **/data** and check the content in the file.

The commands are as follows:

```
ansible Nginx -m command -a "mkdir /data"
ansible Nginx -m command -a "cp /etc/passwd /data"
ansible Nginx -m command -a "removes=/data/passwd cat /data/passwd"
```

```
[root@Ansible ~]# ansible Nginx -m command -a "mkdir /data"
[WARNING]: Consider using the file module with state=directory rather than
command task or set 'command_warnings=False' in ansible.cfg to get rid
192.168.1.14 | CHANGED | rc=0 >>
192.168.1.15 | CHANGED | rc=0 >>

[root@Ansible ~]# ansible Nginx -m command -a "cp /etc/passwd /data"
192.168.1.15 | CHANGED | rc=0 >>
192.168.1.14 | CHANGED | rc=0 >>

[root@Ansible ~]# ansible Nginx -m command -a "removes=/data/passwd cat /data/passwd"
192.168.1.14 | CHANGED | rc=0 >>
root:x:0:root:/root:/bin/bash
bin:x:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
```

- **command** is the default module of Ansible. Can you prove it?

Answer: If the **-m** option is not used when running the Ansible command, the **command** module is invoked.

### 1.3.2 Practice of the shell Module

Requirement: Check whether the **/** directory in the Nginx host group contains directories related to **data**. If yes, use "this is a test" to overwrite the file content in the **data** directory.

The commands are as follows:

```
ansible Nginx -m shell -a "ls / | grep data"
ansible Nginx -m shell -a "ls /data"
ansible Nginx -m shell -a "echo 'this is a test' > /data/passwd"
```

```
[root@Ansible ~]# ansible Nginx -m shell -a "ls / | grep data"
192.168.1.14 | CHANGED | rc=0 >>
data
192.168.1.15 | CHANGED | rc=0 >>
data
```

```
[root@Ansible ~]# ansible Nginx -m shell -a "ls /data"
192.168.1.14 | CHANGED | rc=0 >>
passwd
192.168.1.15 | CHANGED | rc=0 >>
passwd

[root@Ansible ~]# ansible Nginx -m shell -a "echo 'this is a test' > /data/passwd"
192.168.1.15 | CHANGED | rc=0 >>
192.168.1.14 | CHANGED | rc=0 >>

[root@Ansible ~]# ansible Nginx -m shell -a "cat /data/passwd"
192.168.1.15 | CHANGED | rc=0 >>
this is a test
192.168.1.14 | CHANGED | rc=0 >>
this is a test
```

### 1.3.3 Practice of the script Module

Requirement: Print the MAC addresses of all the hosts.

Create the following script on the Ansible controller:

```
#!/bin/bash
ip addr | grep link/ether | awk '{print $2}'
```

Run the following command on the controller to execute the script on all the controlled hosts:

```
ansible all -m script -a "~/mac.sh" | grep stdout | awk '{print $2}' | grep \n
```

```
[root@Ansible ~]# ansible all -m script -a "~/mac.sh" | grep stdout | awk '{print $2}' | grep \n
"00:0c:29:88:a6:a9\r\n",
"00:0c:29:a8:db:82\r\n",
"00:0c:29:59:92:64\r\n",
```

### 1.3.4 Practice of the copy Module

Requirement: Create the **/root/data/copy** file on the controller, copy the file to the **/tmp** directory on the controlled node in the Nginx host group, and enter "hello, openEuler". Enter "hello, world" in **/root/data/copy**, copy the file to the same file in the Nginx group again, and keep the destination file unchanged when the file content differs.

The commands are as follows:

```
mkdir /root/data
touch /root/data/copy
```

```
[root@Ansible ~]# mkdir /root/data
[root@Ansible ~]# touch /root/data/copy
```

```
ansible Nginx -m copy -a "src=/root/data/copy dest=tmp"
```

```
[root@Ansible ~]# ansible Nginx -m copy -a "src=/root/data/copy dest=tmp"
192.168.1.15 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
    "checksum": "da39a3ee5e6b4b0d3255bfef95601890af80709",
    "dest": "./tmp".
```

```
ansible Nginx -m copy -a "content='hello,openEuler' dest=/tmp/copy"
```

```
[root@Ansible ~]# ansible Nginx -m copy -a "content='hello,openEuler' dest=/tmp/copy"
192.168.1.15 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
    "checksum": "5e59286702c03b4efed0609a18021c5c3c203bb0",
    "dest": "/tmp/copy",
    "gid": 0,
    "group": "root",
    "md5sum": "dc42520cd478fda02f6a0d41dc0b8987",
    "mode": "0644".
```

```
[root@Ansible ~]# echo "hello,world" > /root/data/test
ansible Nginx -m copy -a "force=no src=/root/data/copy dest=/tmp"
```

```
[root@Ansible ~]# ansible Nginx -m copy -a "force=no src=/root/data/copy dest=/tmp"
192.168.1.15 | SUCCESS => {
    "changed": false,
    "dest": "/tmp",
    "src": "/root/data/copy"
}
192.168.1.14 | SUCCESS => {
    "changed": false,
    "dest": "/tmp",
    "src": "/root/data/copy"
}
```

### 1.3.5 Practice of the fetch Module

Requirement: Save the **/tmp/copy** file on the Nginx host to the **/tmp** directory on the controller.

The commands are as follows:

```
ansible Nginx -m fetch -a "src=/tmp/copy dest=/tmp/"
```

```
[root@Ansible ~]# ansible Nginx -m fetch -a "src=/tmp/copy dest=/tmp/"  
192.168.1.15 | CHANGED => {  
    "changed": true,  
    "checksum": "5e59286702c03b4efed0609a18021c5c3c203bb0",  
    "dest": "/tmp/192.168.1.15/tmp/copy",  
    "md5sum": "dc42520cd478fda02f6a0d41dc0b8987",  
    "remote_checksum": "5e59286702c03b4efed0609a18021c5c3c203bb0",  
    "remote_md5sum": null  
}  
192.168.1.14 | CHANGED => {  
    "changed": true,  
    "checksum": "5e59286702c03b4efed0609a18021c5c3c203bb0",  
    "dest": "/tmp/192.168.1.14/tmp/copy",  
    "md5sum": "dc42520cd478fda02f6a0d41dc0b8987",  
    "remote_checksum": "5e59286702c03b4efed0609a18021c5c3c203bb0",  
    "remote_md5sum": null  
}
```

After the file is fetched, the structure of the **/tmp** directory is as follows:

```
[root@Ansible ~]# tree /tmp  
/tmp  
└── 192.168.1.14  
    └── tmp  
        └── copy  
└── 192.168.1.15  
    └── tmp  
        └── copy  
  
4 directories, 2 files
```

### 1.3.6 Practice of the file Module

Requirement: Create the **/tmp/file/data** directory in the Nginx host group, set **owner** and **group** to **test:test**, and set the permission to **755**. Create the **test** file in the directory, create a soft link pointing to **/tmp/link** for the **test** file, and delete the **/tmp/file** directory.

The commands are as follows:

```
ansible Nginx -m file -a "path=/tmp/file/data owner=test group=test mode=755 state=directory"
```

```
[root@Ansible ~]# ansible Nginx -a "useradd test"
192.168.1.14 | CHANGED | rc=0 >>
192.168.1.15 | CHANGED | rc=0 >>

[root@Ansible ~]# ansible Nginx -m file -a "path=/tmp/file/data owner=test group=test mode=755 state=directory"
192.168.1.14 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
    "gid": 1000,
    "group": "test",
    "mode": "0755",
    "owner": "test",
    "path": "/tmp/file/data",
    "size": 40,
    "state": "directory",
    "uid": 1000
}
```

```
ansible Nginx -m file -a "path=/tmp/file/data/test state=touch"
```

```
[root@Ansible ~]# ansible Nginx -m file -a "path=/tmp/file/data/test state=touch"
192.168.1.14 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
    "dest": "/tmp/file/data/test",
    "gid": 0,
    "group": "root",
    "mode": "0644"
}
```

```
ansible Nginx -m file -a "src=/tmp/file/data/test dest=/tmp/link state=link"
```

```
[root@Ansible ~]# ansible Nginx -m file -a "src=/tmp/file/data/test dest=/tmp/link state=link"
192.168.1.14 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
    "dest": "/tmp/link",
    "gid": 0,
    "group": "root",
    "mode": "0777"
}
```

```
ansible Nginx -m file -a "path=/tmp/file state=absent"
```

```
[root@Ansible ~]# ansible Nginx -m file -a "path=/tmp/file state=absent"
192.168.1.14 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
    "path": "/tmp/file",
    "state": "absent"
}
192.168.1.15 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
    "path": "/tmp/file",
    "state": "absent"
}
```

- Question: What are the owner and group to which the **test** file belongs? Why?

Answer: In this lab, the owner and group to which the **test** file belongs are **root:root**.

This is because the owner and group are not specified when the file is created. By default, the owner and group that are running the Ansible commands are used.

### 1.3.7 Practice of the archive and unarchive Modules

Requirement: Create files **test1** and **test2** in the **/tmp** directory of the Nginx host group, compress them into **test.bz2**, and delete **test1** and **test2**. Copy the compressed file to the **/tmp** directory of the controller, and decompress the compressed file on 192.168.1.14 to the **/tmp** directory on 192.168.1.15.

The commands are as follows:

```
ansible Nginx -m file -a "path=/tmp/test1 state=touch"
ansible Nginx -m file -a "path=/tmp/test1 state=touch"
ansible Nginx -m archive -a "path=/tmp/test1 format=zip remove=yes dest=/tmp/test.zip"
```

```
[root@Ansible ~]# ansible Nginx -m archive -a "path=/tmp/test1,/tmp/test2 format=bz2 remove=yes dest=/tmp/test.bz2"
192.168.1.15 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "archived": [
        "/tmp/test1",
        "/tmp/test2"
    ],
}
```

```
ansible Nginx -m fetch -a "src=/tmp/test.bz2 dest=/tmp/"
```

```
[root@Ansible ~]# ansible Nginx -m fetch -a "src=/tmp/test.bz2 dest=/tmp/"
192.168.1.14 | CHANGED => {
    "changed": true,
    "checksum": "2f09973063fa1faed4302ddf4c92d562ea76b459",
    "dest": "/tmp/192.168.1.14/tmp/test.bz2",
    "md5sum": "619f31427c657257f1550226e6c6c06b",
    "remote_checksum": "2f09973063fa1faed4302ddf4c92d562ea76b459",
    "remote_md5sum": null
}
192.168.1.15 | CHANGED => {
```

```
ansible 192.168.1.15 -m unarchive -a "src=/tmp/192.168.1.14/tmp/test.bz2 dest=/tmp/"
```

```
[root@Ansible ~]# ansible 192.168.1.15 -m unarchive -a "src=/tmp/192.168.1.14/tmp/test.bz2 dest=/tmp/"
192.168.1.15 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
    "dest": "/tmp/",
    "extract_results": {
        "cmd": [
            "/usr/bin/gtar",
            "--extract",
            ...
        ]
    }
}
```

Check whether the execution is complete. The details are as follows:

```
[root@Ansible ~]# ansible 192.168.1.15 -a "ls /tmp" | grep test
test1
test2
test.bz2
```

## 1.4 Playbook Comprehensive Practice

### 1.4.1 Environment Preparation

To exclude the impact from the remaining configurations on this lab, clear the configurations in sections 1.2 and 1.3. In this lab, Zabbix is installed using a playbook. The Zabbix host name is set to **zabbix\_server**, hosts Nginx1 and Nginx2 are installed as Zabbix clients, and the host names are set to **Nginx-01** and **Nginx-02**.

To enhance your understanding of the playbook, this lab will complete the playbook compilation by segment.

### 1.4.2 Lab Practice

#### Step 1 Change the host name.

In this step, variables are used to change the Zabbix host name to **Zabbix\_server**, and change the host names of Nginx01 and Nginx02 to **Nginx-01** and **Nginx-02**.

Modify the Ansible host file as follows:

```
[Nginx]
192.168.1.14 host=01
192.168.1.15 host=02
[Nginx:vars]
group=Nginx
[Zabbix]
192.168.1.4 host=Server
```

Create an Ansible playbook and enter the following commands:

```
---
- hosts: Zabbix
  remote_user: root
  gather_facts: no

  tasks:
    - name: set hostname for 192.168.1.4
      hostname:
        name={{ host }}

    - hosts: Nginx
      remote_user: root
      gather_facts: no

      tasks:
        - name: set hostname for 192.168.1.14 and 192.168.1.15
          hostname:
            name={{ group }}-{{ host }}
```

After the editing is complete, use **ansible-playbook -C** to run the script and check whether there is an error. If an error is reported, rectify the fault. After no error is reported, perform the following steps.

## Step 2 Install and configure **zabbix-agent2**.

Add the following content to the Ansible playbook and set related tags:

```
.....  
  
tasks:  
    - name: set hostname for 192.168.1.14 and 192.168.1.15  
      hostname:  
        name={{ group }}-{{ host }}  
    - name: download Zabbix yum repolist  
      tags: agent1  
      shell: rpm -Uvh https://repo.zabbix.com/zabbix/6.2/rhel/8/x86_64/zabbix-release-6.2-  
3.el8.noarch.rpm  
    - name: install zabbix-agent  
      tags: agent2  
      yum:  
        name=zabbix-agent2  
        state=present  
    - name: config zabbix-agent  
      tags: agent3  
      replace:  
        path: /etc/zabbix/zabbix_agent2.conf  
        regexp: '^Server=127.0.0.1$'  
        replace: 'Server=192.168.1.4'  
        notify: restart zabbix-agent2  
    - name: config zabbix-agent service  
      tags: agent3  
      service:  
        name: zabbix-agent2  
        state: started  
        enabled: yes  
  
handlers:  
    - name: restart zabbix-agent2  
      service:  
        name: zabbix-agent2  
        state: restarted
```

Use the tag to perform the test step by step until all the items pass the test. Then, proceed to the next step.

## Step 3 Run the playbook.

Run the compiled playbook to install **zabbix-agent**. The details are as follows:

```
[root@Ansible ~]# ansible-playbook zabbix.yml

PLAY [Zabbix] *****

TASK [set hostname for 192.168.1.4] *****
changed: [192.168.1.4]

PLAY [Nginx] *****

TASK [set hostname for 192.168.1.14 and 192.168.1.15] *****
changed: [192.168.1.14]
changed: [192.168.1.15]
```

# 2 Basic Operations of SaltStack

## 2.1 Installing and Configuring SaltStack

### 2.1.1 Installing salt-master and salt-minion

Step 1 Configure a Yum repository.

Log in to all hosts (192.168.1.60 is called SaltStack) and run the following commands to configure the Yum repository required by SaltStack:

```
rpm --import https://repo.saltproject.io/salt/py3/redhat/8/x86_64/latest/SALTSTACK-GPG-KEY.pub
curl -fsSL https://repo.saltproject.io/salt/py3/redhat/8/x86_64/latest.repo | sudo tee
/etc/yum.repos.d/salt.repo
```

The following figure shows the details.

```
[root@Ansible ~]# rpm --import https://repo.saltproject.io/salt/py3/redhat/8/x86_64/latest/SALTSTACK-GPG-KEY.pub
[root@Ansible ~]# curl -fsSL https://repo.saltproject.io/salt/py3/redhat/8/x86_64/latest.repo | sudo tee /etc/yum.repos.d/salt.repo
[salt-latest-repo]
name=Salt repo for RHEL/CentOS 8 PY3
baseurl=https://repo.saltproject.io/salt/py3/redhat/8/x86_64/latest
skip_if_unavailable=True
priority=10
enabled=1
enabled_metadata=1
gpgcheck=1
gpgkey=https://repo.saltproject.io/salt/py3/redhat/8/x86_64/latest/SALTSTACK-GPG-KEY.pub
```

Step 2 Install salt-master.

Run the following command to install salt-master on the host SaltStack:

```
yum install salt-master -y
```

The following figure shows an example.

```
[root@SaltStack ~]# yum install salt-master -y
Salt repo for RHEL/CentOS 8 PY3
OS
everything
EPOL
debuginfo
source
```

After the installation is complete, run the following command to start salt-master and set it to start upon system startup:

```
systemctl enable salt-master --now
```

Run the following command to install salt-ssh. The Zabbix host is managed using SSH, and the Nginx1 and Nginx2 hosts are managed using salt-minion.

```
yum install salt-ssh -y
```

### Step 3 Configure salt-master.

Create the **master.conf** file in the **/etc/salt/master.d** directory on the host SaltStack and enter the following content:

```
interface: 0.0.0.0
publish_port: 4505
ret_port: 4506
pki_dir: /etc/salt/pki/master
file_roots:
    base:
        - /srv/salt/
pillar_roots:
    base:
        - /srv/pillar
```

Restart the salt-master service and check the service status. If any error occurs, rectify the fault before proceeding to the next step.

### Step 4 Install salt-minion.

On Nginx1 and Nginx2, run the following command to install salt-minion:

```
yum install salt-minion -y
```

The following figure shows an example.

```
[root@Zabbix ~]# yum install salt-minion
Salt repo for RHEL/CentOS 8 PY3
Dependencies resolved.
=====
Transaction Summary
=====
Installing:
salt-minion                                     x86_64           3005.1-4.el8
Installing dependencies:
salt                                         x86_64           3005.1-4.el8
=====
Transaction Summary
```

After the installation is complete, run the following command to start salt-minion and set it to start upon system startup:

```
systemctl enable salt-minion --now
```

### Step 5 Configure salt-minion.

Create the **minion.conf** file in the **/etc/salt/minion.d** directory on the host Nginx1 and enter the following content:

```
master: 192.168.1.60
id: Nginx1
```

Create the **minion.conf** file in the **/etc/salt/minion.d** directory on the host Nginx2 and enter the following content:

```
master: 192.168.1.60  
id: Nginx2
```

After the configuration is complete, restart the salt-minion service on the two hosts and ensure that the service is running properly.

## 2.1.2 Configuring the Master and Minion Authentication

Run the **salt-key** command on the master node to check whether the public key sent by the minion is received. If the public key is received, the following information is displayed:

```
[root@SaltStack master.d]# salt-key  
Accepted Keys:  
Denied Keys:  
Unaccepted Keys:  
Nginx1  
Nginx2  
Rejected Keys:
```

Run the **salt-key -A** command to receive all keys. The details are as follows:

```
[root@SaltStack master.d]# salt-key -A  
The following keys are going to be accepted:  
Unaccepted Keys:  
Nginx1  
Nginx2  
Proceed? [n/Y] y  
Key for minion Nginx1 accepted.  
Key for minion Nginx2 accepted.
```

After the keys are received, run the following command to check whether the communication between the master and minion is normal:

```
salt "*" test.ping
```

If the communication is normal, the following information is displayed:

```
[root@SaltStack master.d]# salt "*" test.ping  
Nginx2:  
    True  
Nginx1:  
    True
```

## 2.1.3 Adding a Client in SSH Mode

Requirement: The Zabbix host is managed by the master in SSH mode.

Add the following content about the Zabbix host to **/etc/salt/roster**:

```
Zabbix:  
host: 192.168.1.4  
user: root  
port: 22
```

Modify the SSH configuration file **/etc/ssh/ssh\_config** on the Zabbix host by changing **StrictHostKeyChecking** from **ask** to **no**, as shown in the following figure.

```
# BatchMode no  
# CheckHostIP yes  
# AddressFamily any  
# ConnectTimeout 0  
# StrictHostKeyChecking no  
# IdentityFile ~/.ssh/id_rsa  
# IdentityFile ~/.ssh/id_dsa  
# IdentityFile ~/.ssh/id_ecdsa  
# IdentityFile ~/.ssh/id_ed25519  
# Port 22
```

After the modification, run the **systemctl restart sshd** command to restart the SSH service.

Then, run the following commands to configure SSH mutual trust between the hosts, SaltStack and Zabbix:

```
ssh-keygen  
ssh-copy-id 192.168.1.4
```

After the configuration is complete, run the following command to check whether the communication between the two hosts is normal:

```
salt-ssh Zabbix test.ping
```

Authentication is required for the first connection. Enter the password as prompted. If the connection is normal, the following information is displayed:

```
[root@Ansible salt]# salt-ssh Zabbix test.ping  
Permission denied for host Zabbix, do you want to deploy the salt-ssh key? (password required):  
[Y/n] y  
Password for root@Zabbix:  
Zabbix:  
    True  
[root@Ansible salt]# salt-ssh Zabbix test.ping  
Zabbix:  
    True
```

## 2.2 SaltStack Remote Execution Function Practice

### 2.2.1 Specifying the Target

In this lab, **test.ping** is used to practice multiple methods of specifying targets.

Requirement 1: Check whether the communication between all minions and the master is normal.

Command: **salt '\*' test.ping**

```
[root@SaltStack master.d]# salt '*' test.ping
Nginx2:
    True
Nginx1:
    True
```

Requirement 2: Check whether the communication between all minions whose IDs start with Nginx but do not contain Nginx1 and the master is normal.

Command: **salt 'Nginx[!1]' test.ping**

```
[root@SaltStack master.d]# salt 'Nginx[!1]' test.ping
Nginx2:
    True
```

- Question: Why is the Zabbix host not displayed in Requirement 1?

Answer: Zabbix is not a minion host but a host managed in SSH mode.

- Question: Can grains be used to specify a host?

Answer: Yes. Use the **-G** option when running the command. For example, to test whether the minion whose IP address is **192.168.1.14** communicates with the master, run the **salt -G 'ipv4:192.168.1.14' test.ping** command, as shown in the following figure.

```
[root@SaltStack master.d]# salt -G 'ipv4:192.168.1.14' test.ping
Nginx1:
    True
```

## 2.2.2 Practice of Remote Execution Function Modules – cmd

Run the **ip addr** command for Nginx1 and Nginx2 in list mode. The command is as follows:

```
[root@SaltStack master.d]# salt -L 'Nginx1,Nginx2' cmd.run "ip addr"
```

```
[root@SaltStack master.d]# salt -L 'Nginx1,Nginx2' cmd.run "ip addr"
Nginx1:
  1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
      valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
      valid_lft forever preferred_lft forever
  2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether fa:16:3e:2b:84:91 brd ff:ff:ff:ff:ff:ff
    altname enp0s3
    inet 192.168.1.14/24 brd 192.168.1.255 scope global dynamic noprefixroute ens3
      valid_lft 107992884sec preferred_lft 107992884sec
    inet6 fe80::30cf:d96e:eb24:af85/64 scope link noprefixroute
      valid_lft forever preferred_lft forever
Nginx2:
  1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
      valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
      valid_lft forever preferred_lft forever
  2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether fa:16:3e:2b:84:92 brd ff:ff:ff:ff:ff:ff
    altname enp0s3
    inet 192.168.1.15/24 brd 192.168.1.255 scope global dynamic noprefixroute ens3
      valid_lft 107992884sec preferred_lft 107992884sec
    inet6 fe80::f789:d80d:ea4:4a9a/64 scope link noprefixroute
      valid_lft forever preferred_lft forever
```

- Question: Does **cmd.run** support all commands?

Answer: When **cmd.run** is used, the minion must support the command to be executed. As shown in the following figure, when the **host** command is executed on the minion, the system displays a message indicating that the command is not found.

```
[root@SaltStack master.d]# salt -L 'Nginx1,Nginx2' cmd.run "host"
Nginx2:
  /bin/sh: line 1: host: command not found
Nginx1:
  /bin/sh: line 1: host: command not found
ERROR: Minions returned with non-zero exit code
```

## 2.2.3 Practice of Remote Execution Function Modules – **pkg**

Run the following command to list information about the Yum repository on Nginx1:

```
[root@SaltStack master.d]# salt Nginx1  pkg.list_repos
[WARNING]: No minions returned with non-zero exit code
[root@SaltStack master.d]# salt Nginx1  pkg.list_repos
Nginx1:
  -----
  EPOL:
  -----
  baseurl:
    http://repo.openeuler.org/openEuler-22.03-LTS/EPOL/main/$basearch/
  enabled:
    1
  file:
    /etc/yum.repos.d/openEuler.repo
  gpgcheck:
    1
  gpgkey:
    http://repo.openeuler.org/openEuler-22.03-LTS/OS/$basearch/RPM-GPG-KEY-openEuler
  name:
    EPOL
```

Run the following command to list NGINX of the latest version that can be installed on Nginx1 using the current Yum repository:

```
[root@SaltStack master.d]# salt Nginx1  pkg.available_version nginx
```

```
[root@SaltStack master.d]# salt Nginx1 pkg.available_version nginx
Nginx1:
  1:1.21.5-3.oe2203
```

Run the following command to install NGINX for Nginx1 and Nginx2:

```
[root@SaltStack master.d]# salt 'Nginx*' pkg.install nginx
```

```
1:1.21.5-3.oe2203
[root@SaltStack master.d]# salt 'Nginx*' pkg.install nginx
Nginx1:
-----
gd:
-----
new:
  2.3.2-2.oe2203
old:
gperftools-libs:
-----
new:
  2.9.1-4.oe2203
old:
-----
```

Run the following command to check whether NGINX has been installed:

```
[root@SaltStack master.d]# salt 'Nginx*' cmd.run "nginx -v"
```

```
[root@SaltStack master.d]# salt 'Nginx*' cmd.run "nginx -v"
Nginx1:
  nginx version: nginx/1.21.5
Nginx2:
  nginx version: nginx/1.21.5
```

## 2.2.4 Practice of Remote Execution Function Modules – service

Run the following command to check whether the NGINX service on Nginx1 and Nginx2 is available:

```
[root@SaltStack master.d]# salt 'Nginx*' service.available nginx
```

```
[root@SaltStack master.d]# salt 'Nginx*' service.available nginx
Nginx1:
  True
Nginx2:
  True
```

Run the following command to set the NGINX service on Nginx1 and Nginx2 to automatically start upon system startup:

```
[root@SaltStack master.d]# salt 'Nginx*' service.enable nginx
```

```
[root@SaltStack master.d]# salt 'Nginx*' service.enable nginx
Nginx1:
  True
Nginx2:
  True
```

Run the following command to check whether the automatic startup of NGINX is enabled upon system startup:

```
[root@SaltStack master.d]# salt 'Nginx*' service.enabled Nginx
```

```
[root@SaltStack master.d]# salt 'Nginx*' service.enabled nginx
Nginx1:
    True
Nginx2:
    True
```

Alternatively, run the following command:

```
[root@SaltStack master.d]# salt 'Nginx*' service.disabled Nginx
```

```
[root@SaltStack master.d]# salt 'Nginx*' service.disabled nginx
Nginx1:
    False
Nginx2:
    False
```

Run the following command to start NGINX:

```
[root@SaltStack ~]# salt 'Nginx*' service.start nginx
```

```
[root@SaltStack ~]# salt 'Nginx*' service.start nginx
Nginx2:
    True
Nginx1:
    True
```

## 2.2.5 Practice of Remote Execution Function Modules – network

Run **network.connect** to test whether Nginx1 and Nginx2 can access each other's NGINX service port (80). Before the test, run the following command to obtain the IP addresses of Nginx1 and Nginx2:

```
[root@SaltStack ~]# salt "*" network.ipaddrs
```

```
        ...
[root@SaltStack ~]# salt "*" network.ipaddrs
Nginx2:
    - 192.168.1.15
Nginx1:
    - 192.168.1.14
```

Run the following command to test whether Nginx1 can access the NGINX service of Nginx2:

```
[root@SaltStack ~]# salt Nginx1 network.connect 192.168.1.15 80
```

```
[root@SaltStack ~]# salt Nginx1 network.connect 192.168.1.15 80
Nginx1:
-----
comment:
    Successfully connected to 192.168.1.15 (192.168.1.15) on tcp port 80
result:
    True
```

Use the same method to test whether Nginx2 can access the NGINX service of Nginx1.

```
[root@SaltStack ~]# salt Nginx2 network.connect 192.168.1.14 80
Nginx2:
-----
comment:
    Successfully connected to 192.168.1.14 (192.168.1.14) on tcp port 80
result:
    True
```

## 2.2.6 Practice of Remote Execution Function Modules – file

Requirement: Use SaltStack to create the **/data/Nginx** directory on Nginx1 and Nginx2 as the root directory of the NGINX service, and create **index.html** in the directory as the default page. On this page, the message "hello, *host name*" is displayed, where *host name* indicates the host names of Nginx1 and Nginx2.

Step 1 Create the **/data/Nginx** directory.

Run the following command to create the **/data/Nginx** directory:

```
[root@SaltStack ~]# salt "*" file.mkdir /data/Nginx
Nginx2:
    True
Nginx1:
    True
```

Step 2 Create **index.html**.

Run the following command to create the **index.html** file:

```
[root@SaltStack ~]# salt "*" file.touch /data/Nginx/index.html
Nginx2:
    True
Nginx1:
    True
```

Step 3 Add content to the page.

Run the following command to add content to **index.html**:

```
[root@SaltStack ~]# salt '*' file.append /data/Nginx/index.html "hello, $HOSTNAME"
```

```
[root@SaltStack ~]# salt '*' file.append /data/Nginx/index.html "hello, $HOSTNAME"
Nginx1:
    Wrote 1 lines to "/data/Nginx/index.html"
Nginx2:
    Wrote 1 lines to "/data/Nginx/index.html"
```

#### Step 4 Modify the NGINX configurations.

Run the following command to create the **test.conf** configuration file in the Nginx configuration directory:

```
[root@SaltStack ~]# salt '*' file.touch /etc/nginx/conf.d/test.conf
Nginx1:
    True
Nginx2:
    True
```

Enter the following content in the configuration file:

```
[root@SaltStack ~]# salt '*' file.append /etc/nginx/conf.d/test.conf \
> "server {" \
> "    root /data/Nginx;" \
> "    index index.html;" \
> "}""
[root@SaltStack ~]# salt '*' file.append /etc/nginx/conf.d/test.conf \
> "server {" \
> "    root /data/Nginx;" \
> "    index index.html;" \
> "}"
Nginx1:
    Wrote 4 lines to "/etc/nginx/conf.d/test.conf"
Nginx2:
    Wrote 4 lines to "/etc/nginx/conf.d/test.conf"
```

#### Step 5 Reload the NGINX service.

Run the following command to reload the NGINX service:

```
[root@SaltStack ~]# salt '*' service.reload nginx
Nginx1:
    True
Nginx2:
    True
```

After the preceding operations are complete, check whether the configuration takes effect.

- Question: Is the page displayed as expected?

Answer: No. The host names of Nginx1 and Nginx2 are not displayed on the page. The host name of the master is displayed. The reason is that the master converts variables

and then sends them to the minion. To achieve the expected effect, you are advised to use the Jinja template to invoke the grains value.

## 2.3 SaltStack Configuration Management Function Practice

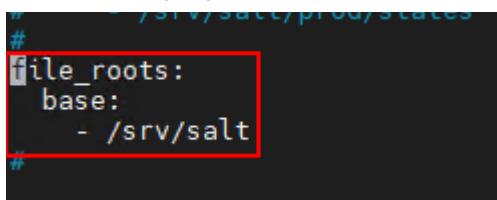
Before starting this lab, delete the NGINX configurations from Nginx1 and Nginx2 to avoid interference.

### 2.3.1 Configuration Management Script Practice

Requirement: The SaltStack management script is used to complete the requirement in section 2.2.6 on Nginx1.

Step 1    Modify the configuration file.

Cancel the configuration of the **file\_roots** parameter on the master node, as shown in the following figure.



```
# /etc/srv/salt/prod/states
#
# file_roots:
#   base:
#     - /srv/salt
```

Create the directory in the configuration.

Run the **systemctl restart salt-master** command to restart the service.

Step 2    Compose the SLS file.

Create the **nginx.sls** file in **/srv/salt** and enter the following content:

```
---
nginx_install:
  pkg.installed:
    - name: nginx

create_test:
  file.touch:
    - name: /etc/nginx/conf.d/test.conf

config_test:
  file.append:
    - name: /etc/nginx/conf.d/test.conf
    - text:
        - "server {"
        - "  root /data/Nginx;"
        - "  index index.html;"
        - "}"
        - ""

create_root_dir:
  file.directory:
```

```
- name: /data/Nginx
- makedirs: True

create_index:
file.touch:
- name: /data/Nginx/index.html

config_index:
file.append:
- name: /data/Nginx/index.html
- text: hello,{{ grains['host'] }}
- template: jinja

enable_nginx:
service.running:
- name: nginx
- enable: True
- reload: True
```

### Step 3 Run the configuration management script.

Run the following command to execute the script compiled in step 2:

```
[root@SaltStack salt]# salt Nginx1 state.sls nginx
```

If the command is successfully executed, the system returns the number of **Succeeded** records, as shown in the following figure.

```
Summary for Nginx1
-----
Succeeded: 7  changed=4 )
Failed:    0
-----
Total states run:      7
Total run time: 146.506 ms
```

- Task: Compile a configuration script to install Apache on Nginx2 and place the home page in **/data/Apache**. The home page content is "hello, *host name*".

Answer: The SLS script for reference is as follows:

```
---
apache_install:
pkg.installed:
- name: httpd

create_test:
file.touch:
- name: /etc/httpd/conf.d/test.conf

config_test:
file.append:
- name: /etc/httpd/conf.d/test.conf
- text:
  - DocumentRoot "/data/apache"
  - '<Directory "/data/apache">'
```

```
- ' AllowOverride None'
- ' Require all granted'
- '</Directory>'

modify_default_config:
file.replace:
- name: /etc/httpd/conf/httpd.conf
- pattern: 'DocumentRoot "/var/www/html"'
- repl: '#DocumentRoot "/var/www/html"'

create_root_dir:
file.directory:
- name: /data/apache
- makedirs: True

create_index:
file.touch:
- name: /data/apache/index.html

config_index:
file.append:
- name: /data/apache/index.html
- text: hello,{{ grains['host'] }}
- template: jinja

enable_apache:
service.running:
- name: httpd
- enable: True
- reload: True
```

### 2.3.2 Composing the top.sls File

Requirement: Install the NGINX service on the minion whose host name is Nginx1, and install the Apache service on the minion whose host name is Nginx2.

Create the **top.sls** file in **/srv/salt** and enter the following content:

```
base:
'host:Nginx1':
- match: grain
- nginx
'host:Nginx2':
- match: grain
- apache
```

Save the file and run the following command to test the **top.sls** file:

```
[root@SaltStack salt]# salt "*" state.highstate test=True
```

```
Summary for Nginx1
-----
Succeeded: 7 (changed=2)
Failed:    0
-----
Total states run:    7
Total run time: 135.717 ms
```

If the test is successful, run the **top.sls** file to complete the requirement.

----- End -----

Huawei openEuler Certification Training

# HCIP-openEuler

## Lab Guide

ISSUE: 1.0



HUAWEI TECHNOLOGIES CO., LTD

**Copyright © Huawei Technologies Co., Ltd. 2024. All rights reserved.**

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

### **Trademarks and Permissions**



HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

### **Notice**

The purchased products, services, and features are stipulated by the commercial contract made between Huawei and the customer. All or partial products, services, and features described in this document may not be within the purchased scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

## **Huawei Technologies Co., Ltd.**

Address: Huawei Industrial Base Bantian, Longgang Shenzhen 518129 People's Republic of China

Website: <https://e.huawei.com>

## Huawei Certification System

Huawei Certification is an integral part of the company's Platform + Ecosystem strategy. It supports the development of ICT infrastructure that features Cloud-Pipe-Device synergy. Our certification is always evolving to reflect the latest trends in ICT development. Huawei Certification consists of three categories: ICT Infrastructure Certification, Basic Software & Hardware Certification, and Cloud Platform & Services Certification, making it the most extensive technical certification program in the industry.

Huawei offers three levels of certification: Huawei Certified ICT Associate (HCIA), Huawei Certified ICT Professional (HCIP), and Huawei Certified ICT Expert (HCIE).

Our programs cover all ICT fields and follow the industry's trend of ICT convergence. With our leading talent development system and certification standards, we are committed to fostering new digital ICT talent and building a sound ICT talent ecosystem.

HCIP-openEuler is intended for frontline engineers at Huawei regional offices or representative offices, and other personnel who want to learn openEuler operations and maintenance (O&M) technologies. HCIP-openEuler certification covers common openEuler enterprise service management, openEuler HA cluster architecture, openEuler storage management, openEuler automated O&M, Linux shell scripts, openEuler system security hardening, and openEuler system monitoring.

Huawei certification helps you unlock opportunities to advance your career and take one more step towards the top of the industry.

# Huawei Career Certification

Cloud Platform & Services Domain		Cloud Computing	Cloud Service	Big Data	GaussDB	IoT
<b>Basic Software &amp; Hardware Domain</b>		openEuler	openGauss	HarmonyOS	AI	Kunpeng
IT Infrastructure Domain	Storage	Collaboration	Intelligent Vision	Digital Power	MDC	
Datacom Domain	Datacom		WLAN		Security	
Optical Domain	Transmission			Access		
Wireless Domain	5G			LTE		

 HCIE  
Huawei Certified ICT Expert

 HCIP  
Huawei Certified ICT Professional

 HCIA  
Huawei Certified ICT Associate

# About This Document

---

## Overview

This document is an HCIP-openEuler certification training course and is intended for trainees who are going to take the HCIP-openEuler exam or readers who want to learn how to build enterprise services, shell scripts, or perform automated O&M using Zabbix or SaltStack on openEuler and other Linux distributions.

## Description

This lab guide consists of six labs, including advanced process control statements, functions, regular expressions, global regular expression search and print (grep) commands, stream editor (sed), and AWK statements in shell scripts.

- Lab 1: Advanced process control statements. Through user password verification, readers can master the process control statements and condition testing.
- Lab 2: Functions. By creating the addUser function, readers can master how to create and use functions.
- Lab 3: Regular expressions. By filtering and querying the NIC and process information, readers can master the usage of regular expression common characters, common metacharacters, and extended metacharacters.
- Lab 4: grep commands. By filtering and querying the NIC and process information, readers can master output control options, context control options, and pattern selection options of grep commands.
- Lab 5: sed. By filtering and querying the NIC and process information, readers can master sed line number addressing, regular expression addressing, adding, deleting, modifying, and querying, search and replacement, file operations, and parameter options.
- Lab 6: AWK statements. By filtering and querying the NIC and process information, readers can master the built-in variables, field separators, record separators, formatted output, regular expressions, control statements, and the script mode of AWK statements.

## Background Knowledge Required

This course is for Huawei certification. To better understand this course, the intended audiences are advised to meet the following requirement:

- Have basic Linux knowledge. You are advised to complete HCIA-openEuler learning and pass the HCIA-openEuler certification exam.

## Lab Environment Overview

In this lab environment, one Elastic Cloud Server (ECS) is required.

## Lab Environment Preparation

### Checking Devices

Before labs start, trainees of each group should check whether the devices in the following table are available:

Device Name	Specifications	Remarks
ECS	1 vCPU   1 GiB   s7.small.1	

# Contents

---

<b>About This Document .....</b>	<b>3</b>
Overview .....	3
Description .....	3
Background Knowledge Required .....	3
Lab Environment Overview.....	4
<b>1 Advanced Process Control Statement Practice.....</b>	<b>1</b>
1.1 Verifying Input.....	1
1.2 Verifying Files .....	4
1.3 Verifying the User Name and Password.....	5
<b>2 Function Practice .....</b>	<b>7</b>
2.1 addUser Function .....	7
<b>3 Regular Expression Practice .....</b>	<b>8</b>
3.1 Common Characters.....	8
3.2 Common Metacharacters.....	8
3.3 Extended Metacharacters.....	9
<b>4 grep Command Practice.....</b>	<b>11</b>
4.1 Output Control Options .....	11
4.2 Context Control Options.....	12
4.3 Pattern Selection Options .....	13
<b>5 sed Practice .....</b>	<b>14</b>
5.1 Line Number Addressing .....	14
5.2 Regular Expression Addressing.....	15
5.3 Adding, Deleting, Modifying, and Querying Texts .....	15
5.4 Searching and Replacing Texts.....	17
5.5 File Operations .....	19
5.6 Parameter Options.....	20
<b>6 AWK Statements .....</b>	<b>22</b>
6.1 Built-in Variables .....	22
6.2 Field Separators .....	23
6.3 Record Separators.....	24
6.4 Formatted Output.....	24
6.5 Regular Expressions.....	25
6.6 Control Statements.....	25
6.7 Script Mode .....	26

# 1

# Advanced Process Control Statement Practice

---

## 1.1 Verifying Input

Step 1 Create a script file.

Log in to the openEuler operating system (OS) and run the following command to create a shell script:

```
[root@openEuler ~]# vim user.sh
```

Specify an interpreter for the script file in **user.sh**.

```
#!/bin/bash
```

Step 2 Add variables.

Add the user name and password variables to the **user.sh** file for verification.

```
#!/bin/bash

username_file="root"
password_file="123456"
```

Step 3 Add the while loop.

The code for adding the while loop to **user.sh** is as follows:

```
#!/bin/bash

username_file="root"
password_file="123456"
echo "Please enter username:"
while :
do
    read username
done
```

Step 4 If the input is empty, continue the input.

Add the if condition for checking whether the input is empty to the while loop.

```
#!/bin/bash

username_file="root"
password_file="123456"
echo "Please enter username:"
while :
do
    read username
    if test -z $username
    then
        echo "Please enter a non empty username:"
        continue
    fi
done
```

Step 5 If the input is correct, the loop exits.

If the input is correct, the while loop exits.

```
#!/bin/bash

username_file="root"
password_file="123456"
echo "Please enter username:"
while :
do
    read username
    if test -z $username
    then
        echo "Please enter a non empty username:"
        continue
    elif test $username = $username_file
    then
        echo "User $username wants to login!"
        break
    fi
done
```

Step 6 If the input is incorrect, continue the input.

If the input is incorrect, continue the while loop.

```
#!/bin/bash

username_file="root"
password_file="123456"
echo "Please enter username:"
while :
do
    read username
    if test -z $username
    then
        echo "Please enter a non empty username:"
        continue
    elif test $username = $username_file
```

```
    then
        echo "User $username wants to login!"
        break
    else
        echo "Incorrect! Please re-enter the username:"
        continue
    fi
done
```

### Step 7 Adding password verification.

After the user name is verified, enter the password and verify it.

```
#!/bin/bash

username_file="root"
password_file="123456"
echo "Please enter username:"
while :
do
    read username
    if test -z $username
    then
        echo "Please enter a non empty username:"
        continue
    elif test $username = $username_file
    then
        echo "User $username wants to login!"
        break
    else
        echo "Incorrect! Please re-enter the username:"
        continue
    fi
done

echo "Please enter password:"
while :
do
    read password
    if test -z $password
    then
        echo "Please enter a non empty password:"
        continue
    elif test $password -eq $password_file
    then
        echo "Welcome user $username login!"
        break
    else
        echo "Incorrect! Please re-enter the password:"
        continue
    fi
done
```

### Step 8 Verify the result.

Save the script and exit the vim insert mode. Enter an empty user name and password, an incorrect user name and password, and a correct user name and password. The following shows an example:

```
[root@openEuler ~]# sh user.sh
Please enter username:
root
User root wants to login!
Please enter password:
a
user.sh: line 29: test: a: integer expression expected
Incorrect! Please re-enter the password:
123456
Welcome user root login!
[root@openEuler ~]#
```

- Question: Why is an alarm generated during script execution? How to modify the script to clear the alarm?

Answer: In this script, **test** compares numbers. Therefore, the entered value must be an integer, and "a" is a character. As a result, a system alarm is generated. To clear the alarm, replace **-eq** in the shell script with an equal sign (=).

## 1.2 Verifying Files

### Step 1 Verifying Files

Add file verification to **user.sh**. The **user.txt** file stores the user name and password. If the **user.txt** file does not exist, the system prompts you to enter the user name and password and saves them to a **user.txt** file.

```
#!/bin/bash

while :
do
    if test -e user.txt
    then
        echo "Entering the Login Program:"
        break
    else
        echo "The user does not exist. Enter the user registration program."
        echo "Please enter username for storage:"
        read username
        echo "Please enter password for storage:"
        read password
    fi
done

username_file="root"
password_file="123456"
```

.....

### Step 2 Check the user name and password based on multiple conditions.

Add the user name and password judgment.

```
#!/bin/bash

while :
do
    if test -e user.txt
    then
        echo "Entering the Login Program:"
        break
    else
        echo "The user does not exist. Enter the user registration program."
        echo "Please enter username for storage:"
        read username
        echo "Please enter password for storage:"
        read password
        if test -n $username -a -n $password
        then
            echo "$username $password" > user.txt
            echo "Storage Successed!"
            echo ""
            break
        else
            echo "Please enter a non empty username or password:"
            continue
        fi
    done
.......
```

## 1.3 Verifying the User Name and Password

### Step 1 Save the user name and password.

Save the data in the **user.txt** file to the user name and password variables for subsequent user name and password verification.

```
username_file=`awk 'NR==1{print $1}' user.txt`
password_file=`awk 'NR==1{print $2}' user.txt`
```

### Step 2 Verify the result.

Save and exit the vim insert mode. Enter an empty user name and password, an incorrect user name and password, and a correct user name and password. The following shows an example:

```
[root@openEuler ~]# sh user.sh
The user does not exist. Enter the user registration program.
```

```
Please enter username for storage:
```

```
root
```

```
Please enter password for storage:
```

```
123
```

```
Storage Successed!
```

```
Please enter username:
```

```
root
```

```
User root wants to login!
```

```
Please enter password:
```

```
123
```

```
Welcome user root login!
```

```
[root@openEuler ~]#
```

# 2 Function Practice

## 2.1 addUser Function

Step 1 Create a function.

Add a function to **user.sh** for adding the user name and password using commands.

```
#!/bin/bash

addUser(){
    if test -n $1 -a -n $2
    then
        echo "$1 $2" >> user.txt
    else
        echo "Add failed.The parameter is incorrect."
    fi
}
```

Step 2 Verify the result.

Run the **source** command to add the **addUser** function to the current shell environment. Run **set** to view the function. Execute the **addUser** function.

```
[root@openEuler ~]# source user.sh
Entering the Login Program:
Please enter username:
^C
[root@openEuler ~]# set
addUser ()
{
    if test -n $1 -a -n $2;
    then
        echo "$1 $2" >> user.txt;
    else
        echo "Add failed.The parameter is incorrect.";
    fi
}
[root@openEuler ~]# addUser a 123
[root@openEuler ~]# cat user.txt
root 123
a 123
[root@openEuler ~]#
```

# 3 Regular Expression Practice

## 3.1 Common Characters

Step 1 Filter by letter.

View the IP addresses of the host.

```
[root@openEuler ~]# ifconfig | grep 'inet'
    inet 192.168.0.22 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::f816:3eff:fe5d:b5cb prefixlen 64 scopeid 0x20<link>
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
[root@openEuler ~]#
```

Step 2 Filter by digit.

View the IP address of the host that contains 192.

```
[root@openEuler ~]# ifconfig | grep '192'
    inet 192.168.0.22 netmask 255.255.255.0 broadcast 192.168.0.255
[root@openEuler ~]#
```

Step 3 Filter by punctuation.

View the IPv6 address of the host.

```
[root@openEuler ~]# ifconfig | grep '::'
    inet6 fe80::f816:3eff:fe5d:b5cb prefixlen 64 scopeid 0x20<link>
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
[root@openEuler ~]#
```

## 3.2 Common Metacharacters

Step 1 Single character.

View the processes whose IDs start with "10".

```
[root@openEuler ~]# ps -ef | grep '10.'
root      10      2  0 02:03 ?        00:00:00 [rcu_tasks_trace]
root     102      2  0 02:03 ?        00:00:00 [kswapd0]
root     103      2  0 02:03 ?        00:00:00 [kpagecache_limi]
root     105      2  0 02:03 ?        00:00:00 [kthrotld]
```

```
root      106      2  0 02:03 ?          00:00:00 [acpi_thermal_pm]
root      107      2  0 02:03 ?          00:00:00 [kmpath_rdacd]
root      108      2  0 02:03 ?          00:00:00 [kaluad]
root      109      2  0 02:03 ?          00:00:00 [ipv6_addrconf]
root     1052     879  0 02:44 pts/0      00:00:00 grep --color=auto 10.
[root@openEuler ~]#
```

## Step 2 The preceding character appears zero or more times.

View the processes whose names contain zero or more hyphens (-) before "gp".

```
[root@openEuler ~]# ps -ef | grep '_*gp'
root      3      2  0 02:03 ?          00:00:00 [rcu_gp]
root      4      2  0 02:03 ?          00:00:00 [rcu_par_gp]
root    1141     879  0 03:11 pts/0      00:00:00 grep --color=auto *_gp
[root@openEuler ~]#
```

## Step 3 The starting position of the line.

View the lines starting with "ens" in the NIC information.

```
[root@openEuler ~]# ifconfig | grep '^ens'
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
[root@openEuler ~]#
```

## Step 4 The ending position of the line.

View the lines ending with "255" in the NIC information.

```
[root@openEuler ~]# ifconfig | grep '255$'
      inet 192.168.0.22  netmask 255.255.255.0  broadcast 192.168.0.255
[root@openEuler ~]#
```

## Step 5 A group of single characters.

View the NIC whose name contains "ens" and a number.

```
[root@openEuler ~]# ifconfig | grep 'ens[0-9]'
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
[root@openEuler ~]#
```

## 3.3 Extended Metacharacters

### Step 1 The preceding character appears one or more times.

View the lines that contain one or more occurrences of "192" in the NIC information.

```
[root@openEuler ~]# ifconfig | grep -E '(192)+'
      inet 192.168.0.22  netmask 255.255.255.0  broadcast 192.168.0.255
[root@openEuler ~]#
```

## Step 2 The preceding character appears zero or one time.

View the lines that contain zero or more occurrences of "255" after "255." in the NIC information.

```
[root@openEuler ~]# ifconfig | grep -E '255.(255)?'
    inet 192.168.0.22 netmask 255.255.255.0 broadcast 192.168.0.255
    inet 127.0.0.1 netmask 255.0.0.0
[root@openEuler ~]#
```

## Step 3 The or character.

View the lines that contain zero or more occurrences of "255" or "0" after "255." in the NIC information.

```
[root@openEuler ~]# ifconfig | grep -E '255.(255|0)+'
    inet 192.168.0.22 netmask 255.255.255.0 broadcast 192.168.0.255
    inet 127.0.0.1 netmask 255.0.0.0
[root@openEuler ~]#
```

## Step 4 Specify the number of repetitions of preceding characters in the NIC information.

View the lines that contain two occurrences of "255." after "255.".

```
[root@openEuler ~]# ifconfig | grep -E '255.(255.){2}'
    inet 192.168.0.22 netmask 255.255.255.0 broadcast 192.168.0.255
[root@openEuler ~]#
```

View the lines that contain two or more occurrences of "255." after "255.".

```
[root@openEuler ~]# ifconfig | grep -E '255.(255.){2,}'
    inet 192.168.0.22 netmask 255.255.255.0 broadcast 192.168.0.255
[root@openEuler ~]#
```

View the lines that contain a maximum of two occurrences of "255." after "255.".

```
[root@openEuler ~]# ifconfig | grep -E '255.(255.){,2}'
    inet 192.168.0.22 netmask 255.255.255.0 broadcast 192.168.0.255
    inet 127.0.0.1 netmask 255.0.0.0
[root@openEuler ~]#
```

View the lines that contain one to two occurrences of "255." after "255.".

```
[root@openEuler ~]# ifconfig | grep -E '255.(255.){1,2}'
    inet 192.168.0.22 netmask 255.255.255.0 broadcast 192.168.0.255
[root@openEuler ~]#
```

# 4 grep Command Practice

## 4.1 Output Control Options

Step 1 A maximum of *n* lines are displayed.

View a maximum of one line that contains "inet" in the NIC information.

```
[root@openEuler ~]# ifconfig | grep -m1 'inet'  
      inet 192.168.0.22 netmask 255.255.255.0 broadcast 192.168.0.255  
[root@openEuler ~]#
```

Step 2 The matched lines and line numbers are displayed.

View the lines that contain "inet6" and their line numbers in the NIC information.

```
[root@openEuler ~]# ifconfig | grep -n 'inet6'  
3:      inet6 fe80::f816:3eff:fe5d:b5cb  prefixlen 64  scopeid 0x20<link>  
12:      inet6 ::1  prefixlen 128  scopeid 0x10<host>  
[root@openEuler ~]#
```

Step 3 The number of matched lines is displayed.

View the number of lines that contain "inet6" in the NIC information.

```
[root@openEuler ~]# ifconfig | grep -c 'inet6'  
2  
[root@openEuler ~]#
```

Step 4 The unmatched lines are displayed.

View the lines that do not start with a space in the NIC information.

```
[root@openEuler ~]# ifconfig | grep -v '^ '  
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500  
  
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536  
  
[root@openEuler ~]#
```

Step 5 Only the matched content is displayed.

View the content that only contains "192.168.0.255" in the NIC information.

```
[root@openEuler ~]# ifconfig | grep -o '192.168.0.255'
```

```
192.168.0.255  
[root@openEuler ~]#
```

#### Step 6 The matched lines and file names are displayed.

View the lines and file names that contain "Ethernet" in the NIC configuration file.

```
[root@openEuler ~]# grep -H 'Ethernet' /etc/sysconfig/network-scripts/ifcfg-ens3  
/etc/sysconfig/network-scripts/ifcfg-ens3:TYPE=Ethernet  
[root@openEuler ~]#
```

#### Step 7 The matched file names are displayed.

View the file names that contain "Ethernet" in the NIC configuration file.

```
[root@openEuler ~]# grep -l 'Ethernet' /etc/sysconfig/network-scripts/ifcfg-ens3  
/etc/sysconfig/network-scripts/ifcfg-ens3  
[root@openEuler ~]#
```

## 4.2 Context Control Options

#### Step 1 The last $n$ lines that are matched.

View the lines that contain "192" and the following one line in the NIC information.

```
[root@openEuler ~]# ifconfig | grep -A1 '192'  
    inet 192.168.0.22 netmask 255.255.255.0 broadcast 192.168.0.255  
    inet6 fe80::f816:3eff:fe5d:b5cb prefixlen 64 scopeid 0x20<link>  
[root@openEuler ~]#
```

#### Step 2 The first $n$ lines that are matched.

View the lines that contain "192" and one line before the matched line in the NIC information.

```
[root@openEuler ~]# ifconfig | grep -B1 '192'  
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.0.22 netmask 255.255.255.0 broadcast 192.168.0.255  
[root@openEuler ~]#
```

#### Step 3 The matched line and $n$ lines before and after it.

View the lines that contain "192" and one line before and after the matched line in the NIC information.

```
[root@openEuler ~]# ifconfig | grep -C1 '192'  
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.0.22 netmask 255.255.255.0 broadcast 192.168.0.255  
    inet6 fe80::f816:3eff:fe5d:b5cb prefixlen 64 scopeid 0x20<link>  
[root@openEuler ~]#
```

## 4.3 Pattern Selection Options

Step 1 Match only whole words.

View the lines that match the whole word "inet" in the NIC information.

```
[root@openEuler ~]# ifconfig | grep -w 'inet'  
    inet 192.168.0.22 netmask 255.255.255.0 broadcast 192.168.0.255  
    inet 127.0.0.1 netmask 255.0.0.0  
[root@openEuler ~]#
```

Step 2 Match only whole lines.

View the lines that match the whole line "TYPE=Ethernet" in the NIC configuration file.

```
[root@openEuler ~]# grep -x 'TYPE=Ethernet' /etc/sysconfig/network-scripts/ifcfg-ens3  
TYPE=Ethernet  
[root@openEuler ~]#
```

Step 3 Ignore case distinctions.

View the lines that match "broadcast" and ignore case distinctions in the NIC information.

```
[root@openEuler ~]# ifconfig | grep -i 'broadcast'  
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
        inet 192.168.0.22 netmask 255.255.255.0 broadcast 192.168.0.255  
[root@openEuler ~]#
```

# 5 sed Practice

## 5.1 Line Number Addressing

Step 1 Match a single line.

```
[root@openEuler ~]# ifconfig | grep -n "
1:ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
2:      inet 192.168.0.22  netmask 255.255.255.0  broadcast 192.168.0.255
3:      inet6 fe80::f816:3eff:fe5d:b5cb  prefixlen 64  scopeid 0x20<link>
4:      ether fa:16:3e:5d:b5:cb  txqueuelen 1000  (Ethernet)
5:      RX packets 128447  bytes 55452425 (52.8 MiB)
6:      RX errors 0  dropped 0  overruns 0  frame 0
7:      TX packets 119518  bytes 13028554 (12.4 MiB)
8:      TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
9:
10:lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
11:      inet 127.0.0.1  netmask 255.0.0.0
12:      inet6 ::1  prefixlen 128  scopeid 0x10<host>
13:      loop  txqueuelen 1000  (Local Loopback)
14:      RX packets 0  bytes 0 (0.0 B)
15:      RX errors 0  dropped 0  overruns 0  frame 0
16:      TX packets 0  bytes 0 (0.0 B)
17:      TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
18:
[root@openEuler ~]#
```

View line 2 in the NIC information.

```
[root@openEuler ~]# ifconfig | sed -n '2p'
inet 192.168.0.22  netmask 255.255.255.0  broadcast 192.168.0.255
[root@openEuler ~]#
```

Step 2 Match multiple specified lines.

View lines 2 to 3 in the NIC information.

```
[root@openEuler ~]# ifconfig | sed -n '2,3p'
inet 192.168.0.22  netmask 255.255.255.0  broadcast 192.168.0.255
inet6 fe80::f816:3eff:fe5d:b5cb  prefixlen 64  scopeid 0x20<link>
[root@openEuler ~]#
```

Step 3 Match lines by the specified step.

Print from line 1 with a step of 2 in the NIC information.

```
[root@openEuler ~]# ifconfig | sed -n '1~2p'
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet6 fe80::f816:3eff:fe5d:b5cb prefixlen 64 scopeid 0x20<link>
        RX packets 128587 bytes 55463781 (52.8 MiB)
        TX packets 119617 bytes 13039916 (12.4 MiB)

        inet 127.0.0.1 netmask 255.0.0.0
        loop txqueuelen 1000 (Local Loopback)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
[root@openEuler ~]#
```

Step 4 Match *n* lines after the specified line.

View line 2 and the following three lines in the NIC information.

```
[root@openEuler ~]# ifconfig | sed -n '2,+3p'
    inet 192.168.0.22 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::f816:3eff:fe5d:b5cb prefixlen 64 scopeid 0x20<link>
        ether fa:16:3e:5d:b5:cb txqueuelen 1000 (Ethernet)
        RX packets 128614 bytes 55465947 (52.8 MiB)
[root@openEuler ~]#
```

## 5.2 Regular Expression Addressing

Step 1 Match lines using common regular expressions.

View the lines that contain "192" in the NIC information.

```
[root@openEuler ~]# ifconfig | sed -n '/192/p'
    inet 192.168.0.22 netmask 255.255.255.0 broadcast 192.168.0.255
[root@openEuler ~]#
```

Step 2 Match lines using extended regular expressions.

View the lines that contain six consecutive digits in the NIC information.

```
[root@openEuler ~]# ifconfig | sed -rn '/[0-9]{6}/p'
    RX packets 129004 bytes 55498555 (52.9 MiB)
    TX packets 119919 bytes 13071368 (12.4 MiB)
[root@openEuler ~]#
```

## 5.3 Adding, Deleting, Modifying, and Querying Texts

Step 1 Insert before specified lines.

Insert one line before the first line in the NIC configuration information.

```
[root@openEuler ~]# ifconfig | sed '1iNetwork card configuration information'
Network card configuration information
```

```
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.0.22 netmask 255.255.255.0 broadcast 192.168.0.255
      inet6 fe80::f816:3eff:fe5d:b5cb prefixlen 64 scopeid 0x20<link>
        ether fa:16:3e:5d:b5:cb txqueuelen 1000 (Ethernet)
          RX packets 129196 bytes 55514263 (52.9 MiB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 120085 bytes 13088784 (12.4 MiB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
      loop txqueuelen 1000 (Local Loopback)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[root@openEuler ~]#
```

## Step 2 Insert after specified lines.

Insert one line after the first line in the NIC configuration information.

```
[root@openEuler ~]# ifconfig | sed '1aNetwork card configuration information'
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
Network card configuration information
      inet 192.168.0.22 netmask 255.255.255.0 broadcast 192.168.0.255
      inet6 fe80::f816:3eff:fe5d:b5cb prefixlen 64 scopeid 0x20<link>
        ether fa:16:3e:5d:b5:cb txqueuelen 1000 (Ethernet)
          RX packets 129257 bytes 55519729 (52.9 MiB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 120138 bytes 13094862 (12.4 MiB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
      loop txqueuelen 1000 (Local Loopback)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[root@openEuler ~]#
```

## Step 3 Delete specified lines.

Delete the first line in the NIC configuration information.

```
[root@openEuler ~]# ifconfig | sed '1d'
      inet 192.168.0.22 netmask 255.255.255.0 broadcast 192.168.0.255
      inet6 fe80::f816:3eff:fe5d:b5cb prefixlen 64 scopeid 0x20<link>
        ether fa:16:3e:5d:b5:cb txqueuelen 1000 (Ethernet)
          RX packets 129383 bytes 55530875 (52.9 MiB)
```

```
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 120233 bytes 13105352 (12.4 MiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
            RX packets 0 bytes 0 (0.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 0 bytes 0 (0.0 B)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[root@openEuler ~]#
```

#### Step 4 Replace specified lines.

Replace the first line in the NIC configuration information.

```
[root@openEuler ~]# ifconfig | sed '1cNetwork card configuration information'
Network card configuration information
    inet 192.168.0.22 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::f816:3eff:fe5d:b5cb prefixlen 64 scopeid 0x20<link>
        ether fa:16:3e:5d:b5:cb txqueuelen 1000 (Ethernet)
        RX packets 129453 bytes 55537071 (52.9 MiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 120289 bytes 13111608 (12.5 MiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
            RX packets 0 bytes 0 (0.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 0 bytes 0 (0.0 B)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[root@openEuler ~]#
```

## 5.4 Searching and Replacing Texts

#### Step 1 Replace the first matched content in the search.

Replace the first "255" of each line in the NIC information with "11111111".

```
[root@openEuler ~]# ifconfig | sed 's/255/11111111/'
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.22 netmask 11111111.255.255.0 broadcast 192.168.0.255
    inet6 fe80::f816:3eff:fe5d:b5cb prefixlen 64 scopeid 0x20<link>
        ether fa:16:3e:5d:b5:cb txqueuelen 1000 (Ethernet)
        RX packets 129819 bytes 55567269 (52.9 MiB)
        RX errors 0 dropped 0 overruns 0 frame 0
```

```
TX packets 120546 bytes 13141330 (12.5 MiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 11111111.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[root@openEuler ~]#
```

## Step 2 Search and replace all the matched content.

Replace every "255" in the NIC information with "11111111".

```
[root@openEuler ~]# ifconfig | sed 's/255/11111111/g'
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.22 netmask 11111111.11111111.11111111 broadcast 192.168.0.11111111
    inet6 fe80::f816:3eff:fe5d:b5cb prefixlen 64 scopeid 0x20<link>
        ether fa:16:3e:5d:b5:cb txqueuelen 1000 (Ethernet)
        RX packets 129829 bytes 55568127 (52.9 MiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 120561 bytes 13143500 (12.5 MiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 11111111.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[root@openEuler ~]#
```

## Step 3 Search and replace by using custom separators.

Replace the slash (/) with the at sign (@) for search.

```
[root@openEuler ~]# ifconfig | sed 's@RX@Receive@'
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.22 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::f816:3eff:fe5d:b5cb prefixlen 64 scopeid 0x20<link>
        ether fa:16:3e:5d:b5:cb txqueuelen 1000 (Ethernet)
        Receive packets 130033 bytes 55584699 (53.0 MiB)
        Receive errors 0 dropped 0 overruns 0 frame 0
        TX packets 120713 bytes 13166204 (12.5 MiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
```

```
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
Receive packets 0 bytes 0 (0.0 B)
Receive errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
[root@openEuler ~]#
```

#### Step 4 Reference a search string in a replacement string.

The text matched by the search string "RX" replaces & in the replacement string.

```
[root@openEuler ~]# ifconfig | sed 's/RX/&:Receive/'
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.0.22 netmask 255.255.255.0 broadcast 192.168.0.255
        inet6 fe80::f816:3eff:fe5d:b5cb prefixlen 64 scopeid 0x20<link>
          ether fa:16:3e:5d:b5:cb txqueuelen 1000 (Ethernet)
          RX:Receive packets 130080 bytes 55588485 (53.0 MiB)
          RX:Receive errors 0 dropped 0 overruns 0 frame 0
          TX packets 120746 bytes 13170194 (12.5 MiB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
          loop txqueuelen 1000 (Local Loopback)
          RX:Receive packets 0 bytes 0 (0.0 B)
          RX:Receive errors 0 dropped 0 overruns 0 frame 0
          TX packets 0 bytes 0 (0.0 B)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[root@openEuler ~]#
```

## 5.5 File Operations

#### Step 1 Read content from another file.

Read the content from the NIC configurations, insert the read content after the first line of the **ifconfig1.txt** file, and the **ifconfig1.txt** file is not edited.

```
[root@openEuler ~]# echo 'myifconfig' > ifconfig1.txt
[root@openEuler ~]# sed '1r /etc/sysconfig/network-scripts/ifcfg-ens3' ifconfig1.txt
myifconfig
# Created by cloud-init on instance boot automatically, do not edit.
#
BOOTPROTO=dhcp
DEVICE=ens3
HWADDR=fa:16:3e:5d:b5:cb
ONBOOT=yes
TYPE=Ethernet
USERCTL=no
```

```
[root@openEuler ~]#
```

### Step 2 Save content as another file.

Save the first line of the **ifconfig1.txt** file to the **ifconfig2.txt** file.

```
[root@openEuler ~]# sed '1w ifconfig2.txt' ifconfig1.txt
myifconfig
[root@openEuler ~]# cat ifconfig2.txt
myifconfig
[root@openEuler ~]#
```

## 5.6 Parameter Options

### Step 1 Modify the input file.

Change the first line in **ifconfig2.txt** to "myifconig2".

```
[root@openEuler ~]# sed -i '1c myifconfig2' ifconfig2.txt
[root@openEuler ~]# cat ifconfig2.txt
myifconfig2
[root@openEuler ~]#
```

### Step 2 Execute multiple editing commands.

Print the first line of **ifconfig2.txt** and insert "Network card configuration information" after the first line.

```
[root@openEuler ~]# sed -e '1p' -e '1a Network card configuration information' ifconfig2.txt
myifconfig2
myifconfig2
Network card configuration information
[root@openEuler ~]#
```

### Step 3 Specify the sed script.

Create the **ifconfig2.sed** script and perform the editing operations described in Step 2.

```
[root@openEuler ~]# vim ifconfig2.sed
[root@openEuler ~]# cat ifconfig2.sed
#!/bin/sed -f

1p
1a Network card configuration information
[root@openEuler ~]# sed -f ifconfig2.sed ifconfig2.txt
myifconfig2
myifconfig2
Network card configuration information
[root@openEuler ~]#
```

#### Step 4 Cancel the default output.

Cancel the default output when running commands and display only the changed lines.

```
[root@openEuler ~]# sed -n '1aNetwork card configuration information' ifconfig2.txt
Network card configuration information
[root@openEuler ~]#
```

# 6 AWK Statements

## 6.1 Built-in Variables

### Step 1 Current records.

Print the lines that contain "inet" in the NIC information:

```
[root@openEuler ~]# ifconfig | awk '/inet/{print $0}'  
    inet 192.168.0.22 netmask 255.255.255.0 broadcast 192.168.0.255  
    inet6 fe80::f816:3eff:fe5d:b5cb prefixlen 64 scopeid 0x20<link>  
    inet 127.0.0.1 netmask 255.0.0.0  
    inet6 ::1 prefixlen 128 scopeid 0x10<host>  
[root@openEuler ~]#
```

### Step 2 Record fields.

Print the IP address and subnet mask in the NIC information.

```
[root@openEuler ~]# ifconfig | awk '/inet/{print $2,$4}'  
192.168.0.22 255.255.255.0  
fe80::f816:3eff:fe5d:b5cb 64  
127.0.0.1 255.0.0.0  
::1 128  
[root@openEuler ~]#
```

### Step 3 The number of record fields.

Print the number of columns of the IPv6 address in the NIC information.

```
[root@openEuler ~]# ifconfig | awk '/inet6/{print NF}'  
6  
6  
[root@openEuler ~]#
```

### Step 4 Last column.

Print the IPv4 broadcast address in the NIC information.

```
[root@openEuler ~]# ifconfig | awk '/inet /{print $NF}'  
192.168.0.255  
255.0.0.0  
[root@openEuler ~]#
```

### Step 5 Line number.

Print the last column of the second line.

```
[root@openEuler ~]# ifconfig | awk 'NR==2{print $NF}'  
192.168.0.255  
[root@openEuler ~]#
```

## 6.2 Field Separators

### Step 1 Input field separator (FS).

Print the eight groups of IPv6 addresses separately. Method 1:

```
[root@openEuler ~]# ifconfig | awk '/inet6/{print $2}' > ifconfig3.txt  
[root@openEuler ~]# cat ifconfig3.txt  
fe80::f816:3eff:fe5d:b5cb  
::1  
[root@openEuler ~]# awk -F: '{print $1,$2,$3,$4,$5,$6,$7,$8}' ifconfig3.txt  
fe80 f816 3eff fe5d b5cb  
1  
[root@openEuler ~]#
```

Method 2:

```
[root@openEuler ~]# awk 'BEGIN{FS=":"};{print $1,$2,$3,$4,$5,$6,$7,$8}' ifconfig3.txt  
fe80 f816 3eff fe5d b5cb  
1  
[root@openEuler ~]#
```

### Step 2 Output field separator (OFS).

Print the eight groups of IPv6 addresses that are separated by vertical bars (|). Method 1:

```
[root@openEuler ~]# awk -F: '{print $1"|"$2"|"$3"|"$4"|"$5"|"$6"|"$7"|"$8}' ifconfig3.txt  
fe80||f816|3eff|fe5d|b5cb||  
||1|||||  
[root@openEuler ~]#
```

Method 2:

```
[root@openEuler ~]# awk -F: 'BEGIN{OFS="|"};{print $1,$2,$3,$4,$5,$6,$7,$8}' ifconfig3.txt  
fe80||f816|3eff|fe5d|b5cb||  
||1|||||  
[root@openEuler ~]#
```

## 6.3 Record Separators

Step 1 Input record separator (RS).

Print the IPv4 address by line.

```
[root@openEuler ~]# ifconfig | awk 'NR==2{print $2}' > ifconfig3.txt
[root@openEuler ~]# cat ifconfig3.txt
192.168.0.22
[root@openEuler ~]# awk 'BEGIN{RS="."}{print $0}' ifconfig3.txt
192
168
0
22

[root@openEuler ~]#
```

Step 2 Output record separator (ORS).

Set the ORS to a space.

```
[root@openEuler ~]# awk 'BEGIN{RS=".":ORS=" "};{print $0}' ifconfig3.txt
192 168 0 22
[root@openEuler ~]#
```

## 6.4 Formatted Output

Step 1 print

Output customized IPv4 address information.

```
[root@openEuler ~]# ifconfig | awk 'NR==2{print $0}' > ifconfig3.txt
[root@openEuler ~]# cat ifconfig3.txt
inet 192.168.0.22 netmask 255.255.255.0 broadcast 192.168.0.255
[root@openEuler ~]# awk '{print "ip:"$2,"netmask:"$4,"broadcast:"$6}' ifconfig3.txt
ip:192.168.0.22 netmask:255.255.255.0 broadcast:192.168.0.255
[root@openEuler ~]#
```

Step 2 printf

Output customized IPv4 address information.

```
[root@openEuler ~]# ifconfig | awk 'NR==2;NR==11{print $0}' > ifconfig3.txt
[root@openEuler ~]# cat ifconfig3.txt
inet 192.168.0.22 netmask 255.255.255.0 broadcast 192.168.0.255
inet 127.0.0.1 netmask 255.0.0.0
[root@openEuler ~]# awk '{printf "%-5s%-20s%-10s%-20s%-10s%-20s\n",$1,$2,$3,$4,$5,$6}' ifconfig3.txt
inet 192.168.0.22      netmask    255.255.255.0      broadcast 192.168.0.255
inet 127.0.0.1      netmask    255.0.0.0
[root@openEuler ~]#
```

## 6.5 Regular Expressions

Step 1 Line number judgment.

Print the first two lines of the NIC information.

```
[root@openEuler ~]# ifconfig | awk 'NR<=2{print $0}'  
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500  
          inet 192.168.0.22  netmask 255.255.255.0  broadcast 192.168.0.255  
[root@openEuler ~]#
```

Step 2 Logical judgment.

Print the second column of line 2 or line 11.

```
[root@openEuler ~]# ifconfig | awk 'NR==2 || NR==11 {print $2}'  
192.168.0.22  
127.0.0.1  
[root@openEuler ~]#
```

Step 3 Content judgment.

Print the second column of the first two lines that contain "192".

```
[root@openEuler ~]# ifconfig | awk 'NR<=2 && $0 ~ /192/{print $2}'  
192.168.0.22  
[root@openEuler ~]#
```

## 6.6 Control Statements

Step 1 Condition statements.

Print the IPv4 address in the NIC information.

```
[root@openEuler ~]# ifconfig | awk '{if($1=="inet"){print $2}}'  
192.168.0.22  
127.0.0.1  
[root@openEuler ~]#
```

Step 2 Loop statements.

Add 20 asterisks (\*) before the second column of the second line and 20 number signs (#) after the second column.

```
[root@openEuler ~]# ifconfig | awk 'BEGIN{for(i=1;i<20;i++){printf "*"};printf "\n"};NR==2{print  
$2};END{i=1;while(i<20){printf "#";i++};printf "\n"}'  
*****  
192.168.0.22  
#####  
[root@openEuler ~]#
```

## 6.7 Script Mode

Step 1 Create a script.

Create an AWK script.

```
[root@openEuler ~]# vim ifconfig3.sh
[root@openEuler ~]# cat ifconfig3.sh
#!/bin/awk -f

BEGIN{for(i=1;i<20;i++){printf "*";printf "\n"};NR==2{print $2};END{i=1;while(i<20){printf
="#"";i++};printf "\n"}
[root@openEuler ~]#
```

Step 2 Use the script.

Use the AWK script.

```
[root@openEuler ~]# ifconfig | awk -f ifconfig3.sh
*****
192.168.0.22
#####
[root@openEuler ~]#
```

Huawei openEuler Certification Training

# HCIP-openEuler

## Lab Guide

Issue: 1.0



Huawei Technologies Co., Ltd.

**Copyright © Huawei Technologies Co., Ltd. 2024. All rights reserved.**

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

### **Trademarks and Permissions**



HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

### **Notice**

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

## **Huawei Technologies Co., Ltd.**

Address:       Huawei Industrial Base Bantian, Longgang Shenzhen 518129  
                  People's Republic of China

Website:      <https://e.huawei.com>

## Huawei Certification System

Huawei Certification is an integral part of the company's Platform + Ecosystem strategy. It supports the development of ICT infrastructure that features Cloud-Pipe-Device synergy. Our certification is always evolving to reflect the latest trends in ICT development. Huawei Certification consists of three categories: ICT Infrastructure Certification, Basic Software & Hardware Certification, and Cloud Platform & Services Certification, making it the most extensive technical certification program in the industry.

Huawei offers three levels of certification: Huawei Certified ICT Associate (HCIA), Huawei Certified ICT Professional (HCIP), and Huawei Certified ICT Expert (HCIE).

Our programs cover all ICT fields and follow the industry's trend of ICT convergence. With our leading talent development system and certification standards, we are committed to fostering new digital ICT talent and building a sound ICT talent ecosystem.

Huawei Certified ICT Professor-openEuler (HCIP-openEuler) is intended for frontline engineers at Huawei regional offices or representative offices, and other personnel who want to learn openEuler technologies. HCIP-openEuler certification covers common openEuler enterprise service management, openEuler HA cluster architecture, openEuler storage management, openEuler automated O&M, Linux shell scripts, openEuler system security hardening, and openEuler system monitoring.

Huawei certification helps you unlock opportunities to advance your career and take one more step towards the top of the industry.

## Huawei Career Certification

Cloud Platform & Services		Cloud Computing	Cloud Service	Big Data	GaussDB	IoT
Basic Software & Hardware	Basic Software & Hardware Domain	openEuler	openGauss	HarmonyOS	AI	Kunpeng
IT Infrastructure	IT Infrastructure Domain	Storage	Collaboration	Intelligent Vision	Digital Power	MDC
ICT Infrastructure	Datacom Domain	Datacom	WLAN		Security	Huawei Certified ICT Professional
	Optical Domain	Transmission		Access		HCI
	Wireless Domain	5G		LTE		HCI Associate
Huawei Certified ICT Expert		HCIE				
Huawei Certified ICT Professional		HCIP				
Huawei Certified ICT Associate		HCIA				

# About This Document

---

## Overview

This document is intended for trainees preparing for the HCIP-openEuler certification exam and those who are interested in security technologies of openEuler or other Linux distributions, including security hardening policies, principles and usage of common firewalls, and SELinux mandatory access control policies.

## Description

This document consists of eight labs, covering security hardening policies from five aspects: kernel parameters, authorization and authentication, account and password, file permission, and system services. It also describes the principles and command usage of common firewall tools.

- Lab 1: Kernel parameter hardening
- Lab 2: Authorization and authentication hardening
- Lab 3: Account and password hardening
- Lab 4: File permission hardening
- Lab 5: SSH service hardening
- Lab 6: iptables comprehensive practice
- Lab 7: firewalld comprehensive practice
- Lab 8: SELinux access control practice

## Background Knowledge Required

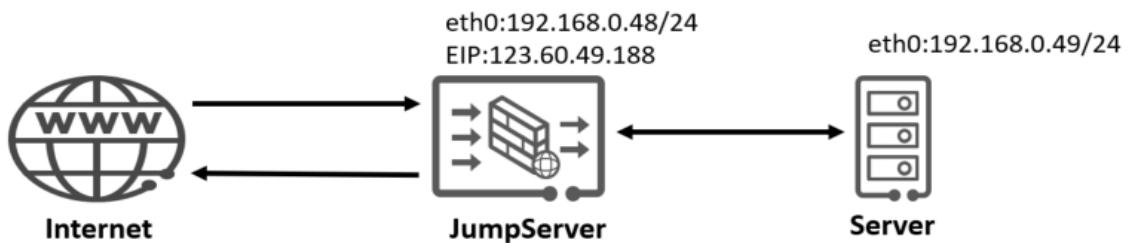
This course is for Huawei's basic certification. To better understand this course, familiarize yourself with the following:

- Linux knowledge. You are advised to complete HCIA-openEuler learning and pass the HCIA-openEuler certification exam.
- Basic network knowledge and the concepts and commands related to the operating system (OS).

## Lab Environment

The lab requires two ECSs with one serving as the bastion host and the other serving as the web application server to security the network. JumpServer is the only entry for accessing intranet services.

---

**Figure 1-1 Topology**

## Environment Preparation

### Checking Devices

Before the lab, each group of trainees should apply for ECSs on Huawei Cloud according to the following table.

Device Name	Specifications	Remarks
JumpServer	2 vCPUs   4 GiB   s7.large.2	
Server	2 vCPUs   4 GiB   s7.large.2	

# Contents

---

<b>About This Document .....</b>	<b>4</b>
Overview .....	4
Description .....	4
Background Knowledge Required .....	4
Lab Environment.....	4
Environment Preparation.....	5
<b>1 OS Security Hardening.....</b>	<b>1</b>
1.1 Introduction.....	1
1.1.1 About This Lab.....	1
1.1.2 Objectives .....	1
1.1.3 Planning .....	1
1.2 Kernel Parameter Hardening Practice .....	1
1.2.1 Hardening Policy Configuration .....	1
1.3 Authorization and Authentication Hardening Practice .....	2
1.3.1 Configuring a Warning for Remote Network Access.....	2
1.3.2 Disabling System Restart Using Ctrl+Alt+Del.....	2
1.3.3 Setting the Automatic Timeout for Terminals .....	3
1.3.4 Setting the Default User's umask Value to 0077 .....	3
1.3.5 Setting the Encryption Password of GRUB2.....	3
1.3.6 Resetting the Root Password.....	4
1.4 Account and Password Hardening Practice .....	5
1.4.1 Shielding System Accounts.....	5
1.4.2 Restricting Accounts that Are Allowed to Use <b>su</b> Commands .....	6
1.4.3 Hardening <b>su</b> Commands .....	6
1.4.4 Setting a Password Validity Period.....	6
1.4.5 Setting Password Complexity .....	6
1.4.6 Setting Password Encryption Algorithms .....	7
1.4.7 Locking an Account After Three Login Failures .....	7
1.5 File Permission Hardening Practice .....	7
1.5.1 Setting File Permissions and Ownership .....	7
1.5.2 Deleting Unowned Files .....	8
1.5.3 Deleting Link Files Pointing to Deleted Files .....	8
1.5.4 Adding the Sticky Bit Property for Globally Writable Directories .....	9
1.5.5 Disabling Global Write on Unauthorized Files.....	9
1.5.6 Restricting Permissions to Run at Commands .....	9
1.5.7 Restricting Permissions to Run cron Commands.....	10

---



1.5.8 Restricting Permissions to Run sudo Commands.....	10
1.6 SSH Service Hardening Practice.....	10
1.6.1 Limiting the Number of Identity Authentication Attempts.....	11
1.6.2 Forbidding Login Using Accounts with Empty Passwords.....	11
1.6.3 Disabling Password-Free Login Based on Trusted Hosts .....	11
1.6.4 Displaying the Date and Time of the Last Login .....	11
1.6.5 Terminating Idle SSH Sessions.....	12
1.6.6 Listening to a Specific Address .....	12
1.6.7 Configuring the SSH Blocklist.....	12
1.6.8 Disabling Login of the Root User.....	12
1.6.9 Allowing Login Through Key Authentication.....	13
1.6.10 Disabling Password Authentication Login.....	14
1.7 Quiz .....	14
<b>2 OS Network Security .....</b>	<b>15</b>
2.1 Introduction .....	15
2.1.1 About This Lab.....	15
2.1.2 Objectives .....	15
2.1.3 Environment Preparation.....	15
2.2 Comprehensive Practices of iptables.....	16
2.2.1 Installing and Enabling iptables .....	16
2.2.2 Configuring JumpServer Rules .....	16
2.2.3 (Optional) Network Address Translation.....	17
2.3 Comprehensive Practices of firewalld.....	20
2.3.1 Installing and Enabling firewalld .....	20
2.3.2 Network Address Translation .....	21
2.3.3 Adding firewalld Rules Using CLI .....	23
2.3.4 Adding firewalld Rules Using an XML File.....	24
2.4 Quiz .....	25
<b>3 Mandatory Access Control on SELinux .....</b>	<b>27</b>
3.1 Introduction .....	27
3.1.1 About This Lab.....	27
3.1.2 Objectives .....	27
3.2 Comprehensive Practices of SELinux Access Control.....	27
3.2.1 Installing and Enabling SELinux .....	27
3.2.2 Restricting SELinux Ports.....	29
3.2.3 Restricting SELinux Types.....	30
3.3 Quiz .....	32

# 1

# OS Security Hardening

---

## 1.1 Introduction

### 1.1.1 About This Lab

This lab describes hardening policies of the openEuler OS from five aspects to improve system and networking security.

### 1.1.2 Objectives

- Master hardening policies for system services.
- Master hardening policies for kernel parameters.
- Master hardening policies for accounts and passwords.
- Master hardening policies for authorization and authentication.
- Master hardening policies for file permissions.

### 1.1.3 Planning

In this lab, the JumpServer cloud host is used as the only entry to the intranet, and security hardening is performed on the host.

## 1.2 Kernel Parameter Hardening Practice

The kernel provides a configurable system control function, which can improve the OS security by controlling various configurable kernel parameters.

- All hardening items are written in the **/etc/sysctl.conf** file.
- Load the **sysctl.conf** file for the configured kernel parameters to take effect.

```
[root@Jumpserver ~]# sysctl -p /etc/sysctl.conf
```

### 1.2.1 Hardening Policy Configuration

Step 1    **ip\_forward**: enables IP routing to facilitate data packet forwarding.

```
net.ipv4.ip_forward=1
```

Step 2    **log\_martians**: records data packets with abnormal IP addresses in logs.

```
net.ipv4.conf.all.log_martians=1
```

Step 3 **send\_redirects**: disables all interfaces from sending IPv4 ICMP redirection packets.

```
net.ipv4.conf.all.send_redirects=0
```

Step 4 **accept\_source\_route**: disables source routing data packets.

```
net.ipv4.conf.all.accept_source_route=0
```

Step 5 **accept\_redirects**: disables the function of accepting ICMP redirection packets.

```
net.ipv4.conf.all.accept_redirects=0
```

Step 6 **icmp\_echo\_ignore\_broadcasts**: disables response to the ping broadcast.

```
net.ipv4.icmp_echo_ignore_broadcasts=1
```

Step 7 **rp\_filter**: disables triangular routing and responds to queries from the same API.  
This prevents IP spoofing.

```
net.ipv4.conf.all.rp_filter=1
```

Step 8 **tcp\_syncookies**: enables SYN flooding protection to prevent DoS attacks.

```
net.ipv4.tcp_syncookies=1
```

## 1.3 Authorization and Authentication Hardening Practice

### 1.3.1 Configuring a Warning for Remote Network Access

The warning is used to warn users before logging in to a system, telling that they may be punished for illegal intrusion to deter potential attackers. In addition, this can also hide system architecture and information to prevent targeted attacks on the system.

You can modify the **/etc/issue.net** file to implement this setting. Replace the original content of the **/etc/issue.net** file with the following information (which has been set in openEuler by default):

```
Authorized users only. All activities may be monitored and reported.
```

### 1.3.2 Disabling System Restart Using Ctrl+Alt+Del

By default, the OS can be restarted by pressing "Ctrl+Alt+Del". You are advised to disable this function to prevent data loss caused by misoperations.

To disable this function, perform the following steps:

Step 1 Delete the two **ctrl-alt-del.target** files.

```
[root@Jumpserver ~]# rm -f /etc/systemd/system/ctrl-alt-del.target  
[root@Jumpserver ~]# rm -f /usr/lib/systemd/system/ctrl-alt-del.target
```

Step 2 Modify the **/etc/systemd/system.conf** file.

Change **#CtrlAltDelBurstAction=reboot-force** to **CtrlAltDelBurstAction=none**.

Step 3 Restart systemd for the modification to take effect.

```
[root@Jumpserver ~]# systemctl daemon-reexec
```

### 1.3.3 Setting the Automatic Timeout for Terminals

Unattended devices are prone to listening or attacks. Therefore, you are advised to set terminals to automatically exit after they stop running for a period of time.

At the end of the **/etc/profile** file, set the **TMOUT** field (unit: second) that specifies the interval for automatic exit as follows:

```
export TMOUT=300
```

### 1.3.4 Setting the Default User's umask Value to 0077

The umask value is used to set the default permission on a newly created file or directory. If the umask value is too small, the permission of group users or other users is too high, which threatens system security. Therefore, the default umask value for all users is set to **0077**. That is, the default permission on directories created by users is **700**, and the default permission on files is **600**.

By default, the default user's umask value in openEuler is **0022**.

Step 1 Add **umask 0077** to the **/etc/bashrc** file and all files in **/etc/profile.d/**.

```
[root@Jumpserver ~]# echo "umask 0077" >> /etc/bashrc  
[root@Jumpserver ~]# for i in `ls /etc/profile.d`;do echo "umask 0077" >> $i;done
```

Step 2 Set the owner and group of the **/etc/bashrc** file and all files in **/etc/profile.d/** to **root**.

**\$FILE** indicates the file name, for example, **/etc/bashrc**.

```
[root@Jumpserver ~]# chown root.root $FILE
```

### 1.3.5 Setting the Encryption Password of GRUB2

The default password of GRUB2 is **openEuler#12**. You are advised to change the default password upon the first login and periodically update the password. If the password is leaked, startup item configurations may be modified, causing the system startup failure.

Step 1 Use the **grub2-mkpasswd-pbkdf2** command to generate an encrypted password.

```
[root@Jumpserver ~]# grub2-mkpasswd-pbkdf2  
Enter password:
```

Reenter password:

PBKDF2 hash of your password is  
grub.pbkdf2.sha512.10000.011476329D0DCD1DE1758B7F19BC275BB60CC6C3DF1082BFB812D249112  
484E7E7C084E6687A957BB91B40F0A8860E78872AACB89F277EB7D68DC2BD86D34E13.B29208FBE27  
1A702ECA14CCB518C165F86564EED56D277434A289948C2504C5AAB73DFCF8E87B160525953DFC51  
A57B3A17CE4452E2DD6BB0FAC1A0F36120FB4

Step 2 Add the following fields to the **/etc/grub2.cfg** file:

```
[root@Jumpserver ~]# vi /etc/grub2.cfg
set superusers="root"
password_pbkdf2 root
grub.pbkdf2.sha512.10000.011476329D0DCD1DE1758B7F19BC275BB60CC6C3DF1082BFB812D249112
484E7E7C084E6687A957BB91B40F0A8860E78872AACB89F277EB7D68DC2BD86D34E13.B29208FBE27
1A702ECA14CCB518C165F86564EED56D277434A289948C2504C5AAB73DFCF8E87B160525953DFC51
A57B3A17CE4452E2DD6BB0FAC1A0F36120FB4
```

The **superusers** field is used to set the account name of the super GRUB2 administrator.

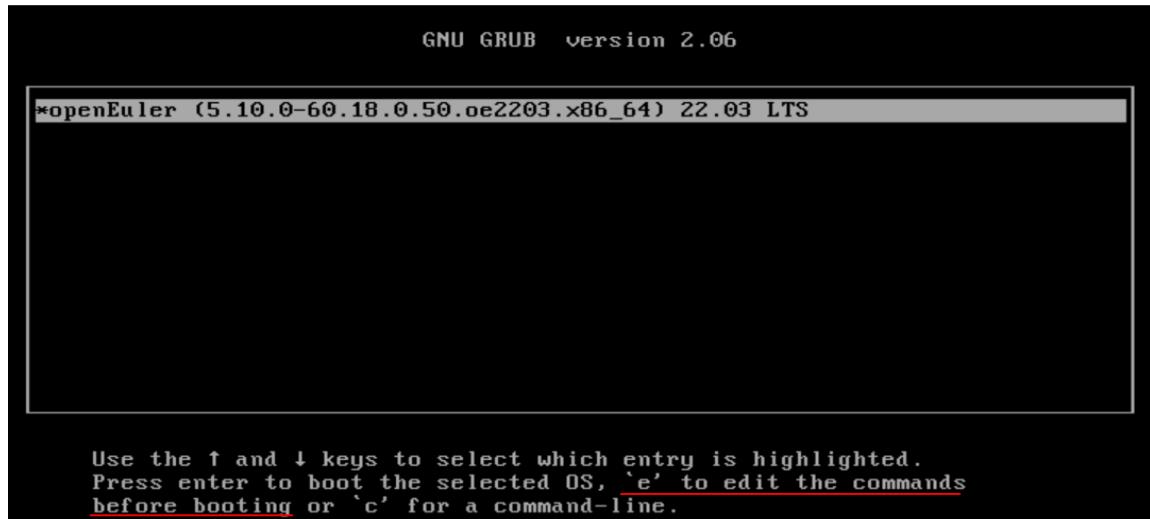
The first parameter following the **password\_pbkdf2** field is the GRUB2 account name, and the second parameter is the encrypted password of the account.

### 1.3.6 Resetting the Root Password

Step 1 Restart the system and go to the GRUB screen.

Restart the openEuler system. When the GUN GRUB screen is displayed, press **e** to modify the startup command of the system.

Enter the user name **root** and the encrypted GRUB2 password to go to the startup command modification screen.



Step 2 Add the **rd.break** parameter to enter the single-user mode.

Find the line starting with "linux" and add **rd.break** to the end of the line.

Press **CTRL+X** to enable the single-user mode.

```
GNU GRUB version 2.06

load_video
set gfxpayload=keep
insmod gzio
insmod fat
search --no-floppy --fs-uuid --set=root 5930-B586
echo      'Loading Linux 5.10.0-60.18.0.50.oe2203.x86_64 ...'
linux    /vmlinuz-5.10.0-60.18.0.50.oe2203.x86_64 root=UUID=ae2\cd454-a1aa-4683-8c9d-711a1d610f40 ro console=tty1 console=ttyS0 rootfstype=\ext4 nomodeset quiet oops=panic softlockup_panic=1 nmi_watchdog=1 rd.shell=\0 selinux=0 crashkernel=256M panic=3 rd.break
echo      'Loading initial ramdisk ...'
initrd   /initramfs-5.10.0-60.18.0.50.oe2203.x86_64.img

Minimum Emacs-like screen editing is supported. TAB lists completions. Press Ctrl-x or F10 to boot, Ctrl-c or F2 for a command-line or ESC to discard edits and return to the GRUB menu.
```

### Step 3 Remount the root directory.

Remount the root directory (`/sysroot`) in writable mode and switch to this environment.

```
sh-5.1# mount -o remount, rw /sysroot
sh-5.1# chroot /sysroot
```

### Step 4 Reset the password of the root user.

```
sh-5.1# echo "Huawei@123" | passwd --stdin root
sh-5.1# exit
sh-5.1# reboot
```

## 1.4 Account and Password Hardening Practice

### 1.4.1 Shielding System Accounts

Accounts are classified into user accounts and system accounts. System accounts can only be used inside a system and cannot be used to log in to the system or perform other operations. Therefore, system accounts are shielded.

Change **shell** of a system account to **/sbin/nologin**. **systemaccount** indicates a system account.

```
usermod -L -s /sbin/nologin systemaccount
```

Find out the accounts that are forbidden from logging in to the system and check the account status.

```
[root@Jumpserver ~]# cat /etc/passwd | grep nologin
bin:x:1:bin:/bin:/sbin/nologin
daemon:x:2:daemon:/sbin:/sbin/nologin
```

```
adm:x:3:4:adm:/var/adm:/sbin/nologin
dhcpd:x:177:177:DHCP server:/sbin/nologin
[root@Jumpserver ~]# passwd -S dhcpcd
dhcpcd LK 2022-03-30 -1 -1 -1 -1 (Password locked.)
```

## 1.4.2 Restricting Accounts that Are Allowed to Use **su** Commands

**su** commands are used to switch between accounts. To enhance system security, it is necessary to control the permissions to use **su** commands. Only the **root** user and those in the **wheel** group are allowed to use **su** commands.

You can modify the **/etc/pam.d/su** file to control the use of **su** commands as follows:

```
auth      required      pam_wheel.so use_uid
```

## 1.4.3 Hardening **su** Commands

To enhance system security and prevent environment variables of the current user from being brought into other environments when you run **su** commands to switch to another user, this configuration has been made in openEuler by default. **PATH** is always initialized when **su** commands are used to switch users.

Implement this setting in the **/etc/login.defs** file as follows:

```
ALWAYS_SET_PATH=yes
```

## 1.4.4 Setting a Password Validity Period

To ensure system security, you are advised to set a password validity period and notify users of changing passwords before the passwords expire.

You can set a password validity period by modifying the hardening-related field in the **/etc/login.defs** file.

```
PASS_MAX_DAYS 90
PASS_MIN_DAYS 0
PASS_MIN_LEN 5
PASS_WARN_AGE 7
```

## 1.4.5 Setting Password Complexity

You can set password complexity by modifying the corresponding configuration file as required.

Add the following content to the first two lines of the password configuration item in the **/etc/pam.d/password-auth** and **/etc/pam.d/system-auth** files:

```
password  requisite  pam_pwquality.so minlen=8 minclass=3 enforce_for_root try_first_pass
local_users_only retry=3 dcredit=0 uccredit=0 lcredit=0 ocredit=0
password  required  pam_pwhistory.so use_authtok remember=5 enforce_for_root
```

## 1.4.6 Setting Password Encryption Algorithms

For system security, passwords cannot be stored in plaintext in the system and must be encrypted. Irreversible cryptographic algorithms must be used in scenarios where passwords do not need to be recovered.

Set the password encryption algorithm to **sha512**, which has been set by default in openEuler. These settings can effectively prevent password disclosure and keep passwords secure.

Modify the **/etc/pam.d/password-auth** and **/etc/pam.d/system-auth** files.

```
password sufficient pam_unix.so sha512 shadow nullok try_first_pass use_authtok
```

## 1.4.7 Locking an Account After Three Login Failures

To ensure system security, you are advised to set the maximum number of incorrect password attempts (three times recommended) and the automatic unlocking time (300 seconds recommended) for a locked account.

During the locking period, any input is considered invalid, but will not cause the locking timer to re-count. Records of users' incorrect attempts are cleared once the account is unlocked. The preceding settings protect passwords from being forcibly cracked and improve system security.

You can set password complexity by modifying the **/etc/pam.d/password-auth** and **/etc/pam.d/system-auth** files. In the following example, the maximum number of incorrect password attempts is set to **3** and the unlocking time after an account is locked is set to **300** seconds.

```
auth required pam_faillock.so preauth audit deny=3 even_deny_root unlock_time=300
auth [default=die] pam_faillock.so authfail audit deny=3 even_deny_root unlock_time=300
auth sufficient pam_faillock.so authsucc audit deny=3 even_deny_root unlock_time=300
```

## 1.5 File Permission Hardening Practice

### 1.5.1 Setting File Permissions and Ownership

Linux treats all objects as files. Even a directory is regarded as a large file containing multiple other files. Therefore, file and directory security is essential to Linux. The security of files and directories is ensured by permissions and ownership.

In openEuler, permissions and ownership for common directories, executable files, and configuration files are set by default.

Take the **/bin** directory as an example. To change the file permission and ownership, perform the following steps:

Step 1    Modify the file permission. For example, set the permission on the **/bin** directory to **755**.

```
chmod 755 /bin
```

- Step 2 Change the file ownership. For example, set the owner and group of the **/bin** directory to **root:root**.

```
chown root:root /bin
```

### 1.5.2 Deleting Unowned Files

When deleting a user or group, the system administrator may forget to delete the files owned by the user or group. If there is a new user or group with an identical name to that of the user or group to be deleted, the new user or group will own some files that do not belong to them. Therefore, you are advised to delete these files.

- Deleting files whose user IDs do not exist

- Step 1 Search for files whose user IDs do not exist.

```
find / -nouser
```

- Step 2 Delete the found files. In the command, **filename** indicates the name of a file whose user ID does not exist.

```
rm -f filename
```

- Deleting files whose group IDs do not exist

- Step 3 Search for files whose group IDs do not exist.

```
find / -nogroup
```

- Step 4 Delete the found files. In the command, **filename** indicates the name of a file whose group ID does not exist.

```
rm -f filename
```

### 1.5.3 Deleting Link Files Pointing to Deleted Files

Link files pointing to deleted files may be exploited by malicious users, deteriorating system security. You are advised to delete those files to improve system security.

After openEuler is installed, link files pointing to deleted files may exist, and these files may have corresponding functions. (Some of them are preconfigured and may be depended on by other components.)

- Step 1 Run the following command to search for link files pointing to deleted files:

```
find dirname -type l -follow 2>/dev/null
```

*dirname* is the name of the directory to be searched. Pay special attention to key directories **/bin**, **/boot**, **/usr**, **/lib64**, **/lib**, and **/var**.

- Step 2 If such files are useless, run the following command to delete them. In the command, **filename** is the name of a link file pointing to deleted files.

```
rm -f filename
```

## 1.5.4 Adding the Sticky Bit Property for Globally Writable Directories

Any user can delete or modify files and directories in globally writable directories. To prevent those files and directories from being arbitrarily deleted, add the sticky bit property for the globally writable directories.

Step 1 Find globally writable directories.

```
find / -type d -perm -0002 ! -perm -1000 -ls | grep -v proc
```

Step 2 Add the sticky bit property for globally writable directories. Replace **dirname** with the actual directory name.

```
chmod +t dirname
```

## 1.5.5 Disabling Global Write on Unauthorized Files

Globally writable files can be modified by any user in a system, which impacts system integrity.

Step 1 Display all globally writable files.

```
for i in `find / -type d -perm -o+w | grep -v proc`;do ls -ld $i | awk '{print $1,$NF}';done  
for i in `find / -type f -perm -o+w | grep -v proc`;do ls -l $i | awk '{print $1,$NF}';done
```

Step 2 View all listed files, exclude files and directories with sticky bits, and delete files or remove their global write permission. In the command, **filename** indicates the corresponding file name.

```
chmod o-w filename
```

## 1.5.6 Restricting Permissions to Run at Commands

**at** commands are used to create tasks that are automatically executed at a specific time point. To prevent users from arbitrarily running **at** commands, you need to specify users who can use the **at** commands. Otherwise, the system may be vulnerable to attacks.

Step 1 Delete the **/etc/at.deny** file.

```
rm -f /etc/at.deny
```

Step 2 Create the **/etc/at.allow** file.

```
touch /etc/at.allow
```

Step 3 Change the owner of the **/etc/at.allow** file to **root:root**.

```
chown root:root /etc/at.allow
```

Step 4 Control the permission on the **/etc/at.allow** file. Only the **root** user can perform this operation.

```
chmod og-rwx /etc/at.allow
```

## 1.5.7 Restricting Permissions to Run cron Commands

**cron** commands are used to create routine tasks. To prevent users from arbitrarily running **cron** commands, you need to specify users who can use the **cron** commands. Otherwise, the system may be vulnerable to attacks.

Step 1 Delete the **/etc/cron.deny** file.

```
rm -f /etc/cron.deny
```

Step 2 Create the **/etc/cron.allow** file.

```
touch /etc/cron.allow
```

Step 3 Change the owner of the **/etc/cron.allow** file to **root:root**.

```
chown root:root /etc/cron.allow
```

Step 4 Control the permission on the **/etc/cron.allow** file. Only the **root** user can perform this operation.

```
chmod og-rwx /etc/cron.allow
```

## 1.5.8 Restricting Permissions to Run sudo Commands

**sudo** commands are used by a common user with the root permission. To enhance system security, it is necessary to control the permission to use **sudo** commands. Only the **root** user is allowed to execute **sudo** commands. By default, openEuler does not restrict the permission of non-root users to execute **sudo** commands.

You can modify the **/etc/sudoers** file to control the use of **sudo** commands. Comment out the following configuration line:

```
#%wheel ALL=(ALL)      ALL
```

## 1.6 SSH Service Hardening Practice

The SSH protocol can effectively prevent information leakage during remote management.

- All hardening items of the SSH service are stored in the **/etc/ssh/sshd\_config** file.

- The configuration that has been commented out in the file is the default openEuler policy.
- After the configuration is modified, restart the SSH service for the modification to take effect.

```
[root@Jumpserver ~]# systemctl restart sshd
```

### 1.6.1 Limiting the Number of Identity Authentication Attempts

It is a good way to mitigate brute force attacks by limiting the maximum number of authentication failures.

```
MaxAuthTries 3
```

If a user fails to enter the correct password for three consecutive times when logging in to the OS, the user will be locked for 60 seconds.

```
Authorized users only. All activities may be monitored and reported.  
openeuler login: test1  
Account temporarily locked due to 3 failed logins  
(48 seconds left to unlock)  
Password:  
Login incorrect
```

### 1.6.2 Forbidding Login Using Accounts with Empty Passwords

Any SSH connection requires a non-null password string.

```
PermitEmptyPasswords no
```

### 1.6.3 Disabling Password-Free Login Based on Trusted Hosts

The **rhosts** file is a method of controlling the trust relationship between systems. If a system trusts another system, the system allows login from the trusted system without a password.

This file is seldom used. It is recommended that you disable it in most cases.

```
IgnoreRhosts yes
```

### 1.6.4 Displaying the Date and Time of the Last Login

By printing the date and time of the last login, users can be aware of unauthorized account login events, which will facilitate investigation of unidentified accesses.

```
PrintLastLog yes
```

The last login time of the user is displayed as follows.

```
WARNING! The remote SSH server rejected X11 forwarding request.  
Welcome to Huawei Cloud Service  
Last failed login: Sun Apr 23 14:23:16 CST 2023 from 119.3.119.20 on ssh:notty  
There were 2 failed login attempts since the last successful login.  
Last login: Sun Apr 23 14:18:19 2023
```

## 1.6.5 Terminating Idle SSH Sessions

Keeping SSH sessions active for a long time may cause security risks. It is recommended that idle SSH sessions be terminated.

```
ClientAliveInterval 900  
ClientAliveCountMax 0
```

Inactive sessions are automatically disconnected after 15 minutes (900 seconds).

```
[root@openeuler ~]# Connection closing...Socket close.  
Connection closed by foreign host.  
Disconnected from remote host(openEuler) at 14:49:12.  
Type `help' to learn how to use Xshell prompt.
```

## 1.6.6 Listening to a Specific Address

By default, SSH listens to all IP addresses configured on the local host, but you should bind SSH to a specific IP address.

```
ListenAddress 192.168.0.48
```

## 1.6.7 Configuring the SSH Blocklist

Users in the **DenyUsers** list are not allowed to log in.

```
DenyUsers $username
```

By default, users not listed in the blocklist are considered trustlist users. Blocklist and trustlist users cannot exist at the same time.

## 1.6.8 Disabling Login of the Root User

Forcing users to remotely log in to the system using specific accounts ensures accountability.

```
PermitRootLogin no
```

The **root** user is not allowed to remotely log in to the system, and a common user is not allowed to switch to the **root** user.

## 1.6.9 Allowing Login Through Key Authentication

Key authentication remembers passwords and allows for password-free authentication.

```
PubkeyAuthentication yes
```

Step 1 Generate a local key.

```
[root@Jumpserver ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:1vI0fXat+Rcu6n8scv0Aiuldq0gSZNJ/AEsQrhaCxLU root@Jumpserver
The key's randomart image is:
+---[RSA 3072]---+
| .+ .
| +. o.
| oooEo
| o..= . . . .
| ..+ . . S + . o o
| . . . o +...o =
| . . . ..o .= .
| . . . = o. =.=.
| oo.= o.o=.*.o
+---[SHA256]---+
```

Step 2 View the key and modify the permission.

```
[root@Jumpserver ~]# ls .ssh/
id_rsa id_rsa.pub known_hosts known_hosts.old
[root@Jumpserver ~]# chmod 600 ~/ssh/id_rsa
[root@Jumpserver ~]# chmod 600 ~/ssh/id_rsa.pub
[root@Jumpserver ~]# chmod 700 ~/ssh
```

Step 3 Upload the public key to the server.

```
[root@Jumpserver ~]# cd .ssh/
[root@Jumpserver .ssh]# ssh-copy-id -i id_rsa.pub root@192.168.0.49
```

Step 4 View the public key and verify the result.

```
[root@Server ~]# ls -l .ssh/
total 12
-rw----- 1 root root 569 May 15 06:45 authorized_keys
-rw----- 1 root root 834 May 15 03:07 known_hosts
-rw-r--r-- 1 root root 94 May 15 03:06 known_hosts.old
```

### 1.6.10 Disabling Password Authentication Login

After a public key is used for login authentication, do not use a password to log in to the system to prevent password leakage. **This policy applies only to the intranet web server.**

PasswordAuthentication no
---------------------------

## 1.7 Quiz

- Can a common user switch to the **root** user after the **root** user is forbidden from remote login during SSHD service hardening?

Answer: Yes.

# 2 OS Network Security

## 2.1 Introduction

### 2.1.1 About This Lab

This lab describes how to use iptables and firewalld, and harden network security based on the lab networking.

### 2.1.2 Objectives

- Master the syntax rules of iptables.
- Master firewalld commands.

### 2.1.3 Environment Preparation

#### 2.1.3.1 Installing the httpd Service on the Server

Step 1     Install the httpd service and enable it to be automatically started upon system startup.

```
[root@Server ~]# dnf install -y httpd
[root@Server ~]# systemctl enable httpd --now
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service →
/usr/lib/systemd/system/httpd.service.
```

Step 2     Check the httpd service status.

```
[root@Server ~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
  Active: active (running) since Tue 2023-05-16 09:30:00 UTC; 7s ago
    Docs: man:httpd.service(8)
   Main PID: 1159 (httpd)
      Status: "Processing requests..."
         Tasks: 177 (limit: 22471)
        Memory: 18.3M
       CGroup: /system.slice/httpd.service
               └─1159 /usr/sbin/httpd -DFOREGROUND
                  ├─1160 /usr/sbin/httpd -DFOREGROUND
                  ├─1161 /usr/sbin/httpd -DFOREGROUND
                  └─1163 /usr/sbin/httpd -DFOREGROUND
```

## 2.2 Comprehensive Practices of iptables

### 2.2.1 Installing and Enabling iptables

Step 1 Install iptables and enable the service.

```
dnf -y install iptables policycoreutils  
service iptables start  
systemctl enable --now iptables.service
```

Step 2 Check the status of iptables.service and enable it to be automatically started upon system startup.

```
[root@Jumpserver ~]# systemctl status iptables  
● iptables.service - IPv4 firewall with iptables  
    Loaded: loaded (/usr/lib/systemd/system/iptables.service; disabled; vendor preset: disabled)  
    Active: active (exited) since Sun 2023-05-14 14:18:27 UTC; 3s ago  
      Process: 1607 ExecStart=/usr/libexec/iptables/iptables.init start (code=exited, status=0/SUCCESS)  
        Main PID: 1607 (code=exited, status=0/SUCCESS)  
[root@Jumpserver ~]# systemctl enable --now iptables.service  
Created symlink /etc/systemd/system/basic.target.wants/iptables.service →  
/usr/lib/systemd/system/iptables.service.
```

### 2.2.2 Configuring JumpServer Rules

Step 1 Clear all rules.

```
iptables -F
```

Step 2 Allow IP addresses from all sources to access TCP port 22 (OpenSSH).

```
iptables -A INPUT -p tcp --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT  
iptables -A OUTPUT -p tcp --sport 22 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

Step 3 Allow the local host to access TCP port 22 (OpenSSH) of all hosts on the intranet (192.168.0.0/24).

```
iptables -A OUTPUT -p tcp -s 192.168.0.0/24 --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j  
ACCEPT  
iptables -A INPUT -p tcp --sport 22 -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

Step 4 Adjust the default policy of a chain.

Following the least privilege principle:

- Forbid all other input traffic by default.
- Forbid the local host to forward data packets by default.
- Forbid hosts to initiate external connections by default. This could effectively defend against attacks such as reverse shell.

```
iptables -P INPUT DROP  
iptables -P FORWARD DROP  
iptables -P OUTPUT DROP
```

**Step 5** Allow the local host to ping other hosts.

```
iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT  
iptables -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT
```

**Step 6** Allow the local host to access UDP port 53 (DNS) of the remote host.

```
iptables -A INPUT -p udp -m udp --sport 53 -j ACCEPT  
iptables -A OUTPUT -p udp -m udp --dport 53 -j ACCEPT
```

**Step 7** Allow for loopback connections.

```
iptables -A INPUT -i lo -j ACCEPT  
iptables -A OUTPUT -o lo -j ACCEPT
```

**Step 8** Allow for established and associated input connections.

```
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

**Step 9** Allow for established output connections.

```
iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED -j ACCEPT
```

**Step 10** View the configured rules and save them.

```
[root@Jumpserver ~]# iptables -nvL --line-numbers  
Chain INPUT (policy DROP 2 packets, 80 bytes)  
num pkts bytes target     prot opt in     out      source          destination  
1   452 28844 ACCEPT     tcp  --  *      *       0.0.0.0/0        0.0.0.0/0        tcp dpt:22 ctstate NEW,ESTABLISHED  
2    0    0 ACCEPT     tcp  --  *      *       0.0.0.0/0        0.0.0.0/0        tcp spt:22 ctstate ESTABLISHED  
3    5   420 ACCEPT     icmp --  *      *       0.0.0.0/0        0.0.0.0/0        icmp type 0  
4    0    0 ACCEPT     all  --  lo     *       0.0.0.0/0        0.0.0.0/0        0.0.0.0/0  
5    7   871 ACCEPT     all  --  *      *       0.0.0.0/0        0.0.0.0/0        ctstate RELATED,ESTABLISHED  
6    0    0 ACCEPT     udp  --  *      *       0.0.0.0/0        0.0.0.0/0        udp spt:53  
  
Chain FORWARD (policy DROP 0 packets, 0 bytes)  
num pkts bytes target     prot opt in     out      source          destination  
  
Chain OUTPUT (policy DROP 0 packets, 0 bytes)  
num pkts bytes target     prot opt in     out      source          destination  
1   340 39764 ACCEPT     tcp  --  *      *       0.0.0.0/0        0.0.0.0/0        tcp spt:22 ctstate ESTABLISHED  
2    0    0 ACCEPT     tcp  --  *      *       192.168.0.0/24  0.0.0.0/0        tcp dpt:22 ctstate NEW,ESTABLISHED  
3    5   420 ACCEPT     icmp --  *      *       0.0.0.0/0        0.0.0.0/0        icmp type 8  
4    0    0 ACCEPT     all  --  *      *       0.0.0.0/0        0.0.0.0/0        ctstate ESTABLISHED  
5    7   483 ACCEPT     udp  --  *      *       0.0.0.0/0        0.0.0.0/0        udp dpt:53
```

```
[root@Jumpserver ~]# service iptables save  
iptables: Saving firewall rules to /etc/sysconfig/iptables: [ OK ]
```

### 2.2.3 (Optional) Network Address Translation

iptables is often used in a network address translation (NAT) environment. NAT can effectively reduce the number of servers with public IP addresses configured and enhance network security.

NAT is classified into source network address translation (SNAT) and destination network address translation (DNAT).

### 2.2.3.1 SNAT

#### Step 1 Disable Source/Destination Check.

On the JumpServer ECS details page, click the **Elastic NICs** tab, unfold it, and set **Source/Destination Check** to **OFF**.

Name	--
NIC ID	ca26231f-6e06-4cd4-b79f-e7c463ed08fa
Status	 Offline
EIP	121.36.99.173   Dynamic BGP   20 Mbit/s
Security Group	HCIP-openEuler
Source/Destination Check	<input checked="" type="checkbox"/> 
IPv4 Subnet ID	15c74c46-21f9-40ee-99b6-8c0f70e14283
IPv6 Subnet ID	--

#### Step 2 Check whether the JumpServer can access the Internet.

```
[root@Jumpserver ~]# ping www.huaweicloud.com
PING koa8myv3.sslego-dk.tcloudscdn.com (110.249.196.149) 56(84) bytes of data.
64 bytes from 110.249.196.149 (110.249.196.149): icmp_seq=1 ttl=51 time=26.6 ms
64 bytes from 110.249.196.149 (110.249.196.149): icmp_seq=2 ttl=51 time=26.7 ms
64 bytes from 110.249.196.149 (110.249.196.149): icmp_seq=3 ttl=51 time=26.6 ms
```

#### Step 3 Enable IP route forwarding of the JumpServer ECS.

```
sysctl -w net.ipv4.ip_forward=1
```

#### Step 4 Configure SNAT and check the configuration.

```
iptables -t filter -A FORWARD -j ACCEPT
iptables -t nat -A POSTROUTING -o ens3 -s 192.168.0.0/24 -j SNAT --to 192.168.0.48
iptables -t nat -nvL --line-numbers
```

#### Step 5 Save iptables configurations and enable them to be automatically started upon system startup.

```
[root@Jumpserver ~]# service iptables save
iptables: Saving firewall rules to /etc/sysconfig/iptables: [ OK ]
[root@Jumpserver ~]# chkconfig iptables on
Note: Forwarding request to 'systemctl enable iptables.service'.
```

```
Created symlink /etc/systemd/system/basic.target.wants/iptables.service →  
/usr/lib/systemd/system/iptables.service.
```

### Step 6 Add a route.

Select a VPC to which a route is to be added and click **Route Tables**. On the **Route Tables** page, click **Add Route**.

Select a property or enter a keyword.					Route Tables
Name/ID	IPv4 CIDR Block	Status	Subnets	Route Tables	
vpc-default 606b0927-ad6d-4cf6-879b-bf671a22f94b	192.168.0.0/16 (Primary ...)	Available	1	1	

Enter the route information as prompted. Set the default gateway of the Server to the private IP address of JmpServer.

Add Route

Route Table rtb-HCIP-openEuler(Default)

Destination Type	Destination	Next Hop Type	Next Hop	Description
IP address	0.0.0.0/0	Server	JmpServer(192.168.0.49)	

### Step 7 Log in to the server and verify the lab result.

```
[root@Server ~]# ping www.huaweicloud.com  
PING koa8myv3.sslego-dk.tcloudscdn.com (110.249.196.149) 56(84) bytes of data.  
64 bytes from 110.249.196.149 (110.249.196.149): icmp_seq=1 ttl=51 time=26.6 ms  
64 bytes from 110.249.196.149 (110.249.196.149): icmp_seq=2 ttl=51 time=26.7 ms  
64 bytes from 110.249.196.149 (110.249.196.149): icmp_seq=3 ttl=51 time=26.6 ms
```

#### 2.2.3.2 DNAT

DNAT enables external users to directly access services provided by servers without external IP addresses.

For example, an external user can access a web application on a server without a public IP address through the Internet (the listening port is TCP 80).

### Step 1 Rewrite the destination address and port.

```
iptables -t nat -A PREROUTING -d 192.168.0.48 -p tcp -m tcp --dport 80 -j DNAT --to-destination 192.168.0.49:80
```

### Step 2 Rewrite the source address.

Change the source IP address to the internal IP address of JumpServer. In this case, the server communicates with JumpServer.

```
iptables -t nat -A POSTROUTING -d 192.168.0.48 -p tcp -m tcp --dport 80 -j SNAT --to-source 192.168.0.49
```

### Step 3 Check and save DNAT configurations.

```
[root@Jumpserver ~]# iptables -t nat -nvL --line-numbers
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
num pkts bytes target prot opt in out source destination
1 2 104 DNAT  tcp -- * * 0.0.0.0/0 192.168.0.48  tcp dpt:80 to:192.168.0.49:80
Chain POSTROUTING (policy ACCEPT 2 packets, 104 bytes)
num pkts bytes target prot opt in out source destination
1 74 4680 SNAT all -- * ens3 192.168.0.0/24 0.0.0.0/0 to:192.168.0.48
2 0 0 SNAT tcp -- * * 0.0.0.0/0 192.168.0.48 tcp dpt:80 to:192.168.0.49
[root@Jumpserver ~]# service iptables save
iptables: Saving firewall rules to /etc/sysconfig/iptables: [ OK ]
```

**Step 4** Access the JumpServer public IP address (<http://123.60.49.188/>) to verify the lab result.



## 2.3 Comprehensive Practices of firewalld

### 2.3.1 Installing and Enabling firewalld

The firewalld service cannot be used together with iptables. Therefore, if iptables, ip6tables, ebtables, and ipset are being used, disable these services before enabling firewalld.

**Step 1** Install firewalld and enable it to be automatically started upon system startup.

```
systemctl disable --now iptables.service
systemctl disable --now ip6tables.service
systemctl disable --now etables.service
systemctl disable --now ipset.service
dnf -y install firewalld firewall-config
systemctl unmask --now firewalld.service
systemctl enable --now firewalld.service
```

**Step 2** Check the firewalld status.

```
[root@Jumpserver ~]# systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor pr
   Active: active (running) since Wed 2016-06-29 14:28:51 CEST; 1 weeks 6 days a
```

```
Docs: man:firewalld(1)
Main PID: 24540 (firewalld)
Tasks: 2 (limit: 512)
CGroup: /system.slice/firewalld.service
└─24540 /usr/bin/python3 -Es /usr/sbin/firewalld --nofork --nrepid
```

## 2.3.2 Network Address Translation

### 2.3.2.1 SNAT

Step 1    Enable route forwarding.

```
[root@Jumpserver ~]# sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

Step 2    Enable address masquerading.

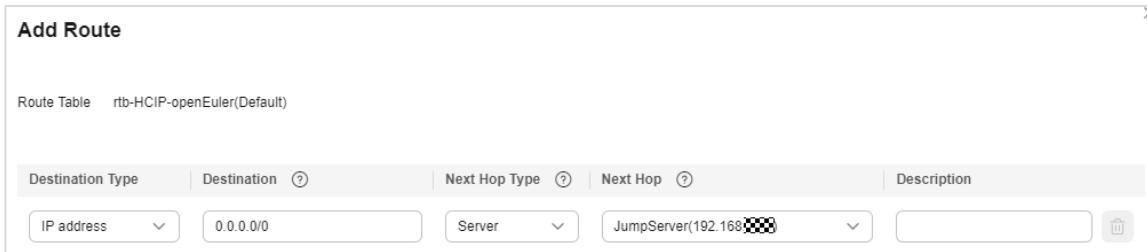
```
[root@Jumpserver ~]# firewall-cmd --add-masquerade
success
[root@Jumpserver ~]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens3
  sources:
  services: dhcpcv6-client mdns ssh
  ports:
  protocols:
  forward: yes
  masquerade: yes
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

Step 3    Add a route.

Select a VPC to which a route is to be added and click **Route Tables**. On the **Route Tables** page, click **Add Route**.

Route Tables					
Name/ID	IPv4 CIDR Block	Status	Subnets	Route Tables	
vpc-default 606b0927-ad6d-4cf6-879b-bf671a22f94b	192.168.0.0/16 (Primary ...)	Available	1	1	

Enter the route information as prompted. Set the default gateway of the Server to the private IP address of JumpServer.



#### Step 4 Log in to the server and verify the lab result.

As shown in the following, the server can access the external network.

```
[root@Jumperver ~]# ssh root@192.168.0.49
[root@Server ~]# ping www.huaweicloud.com -c 3
PING koa8myv3.sslego-dk.tcloudscdn.com (122.189.171.111) 56(84) bytes of data.
64 bytes from 122.189.171.111 (122.189.171.111): icmp_seq=1 ttl=45 time=23.9 ms
64 bytes from 122.189.171.111 (122.189.171.111): icmp_seq=2 ttl=45 time=23.9 ms
64 bytes from 122.189.171.111 (122.189.171.111): icmp_seq=3 ttl=45 time=23.9 ms

--- koa8myv3.sslego-dk.tcloudscdn.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 23.878/23.888/23.907/0.013 ms
```

#### 2.3.2.2 DNAT

##### Step 1 Add port forwarding rules.

```
[root@Jumpserver ~]# firewall-cmd --zone=public --add-forward-
port=port=80:proto=tcp:toport=80:toaddr=192.168.0.86
success
[root@openEuler ~]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens3
  sources:
  services: dhcpcv6-client mdns ssh
  ports:
  protocols:
  forward: yes
  masquerade: yes
  forward-ports:
    port=80:proto=tcp:toport=80:toaddr=192.168.0.86
  source-ports:
  icmp-blocks:
  rich rules:
```

##### Step 2 Access the JumpServer public IP address (<http://123.60.49.188/>) to verify the lab result.



### 2.3.3 Adding firewalld Rules Using CLI

Step 1    Install firewalld and view the default zone information.

```
[root@Server ~]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: eth0
  sources:
  services: ssh mdns dhcpcv6-client
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

In this case, the web service cannot be accessed through <http://123.60.49.188/>.

Step 2    View services supported by firewalld.

By default, firewalld provides more than 80 service groups. Check whether the HTTP service is included.

```
[root@Server ~]# firewall-cmd --get-services | tr " " "\n" | grep 'http'
http
https
wbem-http
wbem-https
```

Step 3    Add the HTTP service to the firewall zone and view the lab result.

```
[root@Server ~]# firewall-cmd --add-service=http
[root@Server ~]# firewall-cmd --list-services
dhcpcv6-client http mdns ssh
```

The web page can be accessed through <http://123.60.49.188/>.

Step 4 Forbid other hosts to ping the local host.

```
[root@Server ~]# firewall-cmd --add-rich-rule='rule family=ipv4 protocol value=icmp reject'  
success
```

Step 5 Set port forwarding rules.

Forward an access request to port 22 of the local host when **192.168.0.209** is used to access port 5555.

```
[root@Server ~]# firewall-cmd --add-rich-rule='rule family=ipv4 source address=192.168.0.209  
forward-port port=5555 protocol=tcp to-port=22'  
success  
[root@Server ~]# firewall-cmd --add-masquerade  
success
```

Step 6 Permanently save firewall configurations.

```
[root@Server ~]# firewall-cmd --runtime-to-permanent  
success
```

### 2.3.4 Adding firewalld Rules Using an XML File

Step 1 Delete firewalld rules.

```
[root@Server ~]# cd /etc/firewalld/zones/  
[root@Server zones]# mv public.xml public.xml.old
```

Step 2 Reload and check firewalld rules.

```
[root@Server zones]# firewall-cmd --reload  
success  
[root@Server zones]# firewall-cmd --list-all  
public (active)  
  target: default  
  icmp-block-inversion: no  
  interfaces: ens3  
  sources:  
  services: dhcpcv6-client mdns ssh  
  ports:  
  protocols:  
  forward: yes  
  masquerade: no  
  forward-ports:  
  source-ports:  
  icmp-blocks:  
  rich rules:
```

Step 3 Use the default firewalld rule file.

```
[root@Server zones]# mv /usr/lib/firewalld/zones/public.xml .
```

#### Step 4 Modify the **public.xml** file as follows:

- (SNAT) Enable address masquerading.
- (DNAT) Forward an access request to port 80 of **192.168.0.86** when an external server is used to access port 80 of the local host.
- Forbid other hosts to ping the local host.
- Forward an access request to port 22 of the local host when **192.168.0.209** is used to access port 5555.

```
<?xml version="1.0" encoding="utf-8"?>
<zone>
    <short>Public</short>
    <description>For use in public areas. You do not trust the other computers on networks to not harm your computer. Only selected incoming connections are accepted.</description>
    <service name="ssh"/>
    <service name="mdns"/>
    <service name="dhcpv6-client"/>
    <masquerade/>
    <forward-port port="80" protocol="tcp" to-port="80" to-addr="192.168.0.86"/>
    <rule family="ipv4">
        <protocol value="icmp"/>
        <reject/>
    </rule>
    <rule family="ipv4">
        <source address="192.168.0.209"/>
        <forward-port port="5555" protocol="tcp" to-port="22"/>
    </rule>
    <forward/>
</zone>
```

#### Step 5 Reload and check rules.

```
[root@Server zones]# firewall-cmd --reload
success
[root@Server zones]# firewall-cmd --list-all
public (active)
    target: default
    icmp-block-inversion: no
    interfaces: ens3
    sources:
    services: dhcpv6-client mdns ssh
    ports:
    protocols:
    forward: yes
    masquerade: yes
    forward-ports:
        port=80:proto=tcp:toport=80:toaddr=192.168.0.86
        source-ports:
        icmp-blocks:
        rich rules:
            rule family="ipv4" protocol value="icmp" reject
            rule family="ipv4" source address="192.168.0.209" forward-port port="5555" protocol="tcp" to-port="22"
```

## 2.4 Quiz

- Can firewalld and iptables be used at the same time?

Answer: No. If iptables is directly used when firewalld is running, some unexpected problems may occur. For example, if a user directly uses iptables to delete rules or chains, the firewall may need to be reloaded to create them again.

- Can a public IP address be directly written during NAT?

Answer: No. The public IP address of Huawei Cloud uses dynamic BGP mapping mode. When writing iptables rules, you must use the private IP address of the host.

# 3

# Mandatory Access Control on SELinux

---

## 3.1 Introduction

### 3.1.1 About This Lab

When SELinux is enabled, the Apache HTTP server (httpd) is restricted by default. Restricted processes run in their own domains and are separated from other restricted processes. If a restricted process is attacked, the attacker's access to resources and possible damage are also restricted due to SELinux policy configurations.

This lab demonstrates the impact of SELinux policies on httpd processes running in their own domains.

### 3.1.2 Objectives

- Be familiar with mandatory access control (MAC) models.
- Master the identification and use of SELinux context information.

## 3.2 Comprehensive Practices of SELinux Access Control

### 3.2.1 Installing and Enabling SELinux

If SELinux has been installed, skip this section.

Step 1    Check the current SELinux mode.

```
[root@Server ~]# getenforce  
Disabled
```

Step 2    Install the SELinux policy.

```
dnf install selinux-policy-targeted, policycoreutils-python-utils
```

Step 3    Configure the SELinux configuration file.

Enable SELinux in permissive mode.

```
[root@Server ~]# vim /etc/selinux/config  
# This file controls the state of SELinux on the system.  
# SELINUX= can take one of these three values:  
#       enforcing - SELinux security policy is enforced.
```

```
#      permissive - SELinux prints warnings instead of enforcing.  
#      disabled - No SELinux policy is loaded.  
SELINUX=permissive  
# SELINUXTYPE= can take one of these two values:  
#      targeted - Targeted processes are protected,  
#      mls - Multi Level Security protection.  
SELINUXTYPE=targeted
```

#### Step 4 Modify system startup parameters.

Check the system boot mode.

```
[root@Server ~]# [ -d /sys/firmware/efi ] && echo UEFI || echo BIOS  
BIOS
```

If **BIOS** is returned, modify the **/etc/grub2.cfg** file. If **UEFI** is returned, modify the **/etc/grub2-efi.cfg** file.

Find the line starting with "linux" in the file and delete **selinux=0**.

```
## BEGIN /etc/grub.d/10_linux ##  
menuentry 'openEuler (5.10.0-60.18.0.50.oe2203.x86_64) 22.03 LTS' --class openEuler --class gnu-linux --class gnu --class os --unrestricted $menuentry_id_option 'gnulinux-5.10.0-60.18.0.50.oe2203.x86_64-advanced-/usr1/mkeuleros/usr1/mkeuleros/result_hmi/system.img' {  
    load_video  
    set gfxpayload=keep  
    insmod gzio  
    insmod fat  
    search --no-floppy --fs-uuid --set=root 5930-B586  
    echo 'Loading Linux 5.10.0-60.18.0.50.oe2203.x86_64 ...'  
    linux /vmlinuz-5.10.0-60.18.0.50.oe2203.x86_64 root=UUID=ae2cd454-a1aa-4683-8c9d-711a1d610f40 ro console=tty1 console=ttyS0 rootfstype=  
ext4 nomodeset quiet oops=panic softlockup_panic=1 nmi_watchdog=1 rd.shell=0 selinux=0 crashkernel=256M panic=3  
    echo 'Loading initial ramdisk ...'  
    initrd /initramfs-5.10.0-60.18.0.50.oe2203.x86_64.img  
}  
## END /etc/grub.d/10_linux ##
```

#### Step 5 Restart the system and check the SELinux mode.

```
[root@Server ~]# getenforce  
Permissive
```

#### Step 6 Re-label the file.

```
[root@Server ~]# fixfiles -F onboot  
System will relabel on next boot
```

#### Step 7 Configure the SELinux configuration file.

Enable SELinux in enforcing mode.

```
[root@Server ~]# vim /etc/selinux/config  
# This file controls the state of SELinux on the system.  
# SELINUX= can take one of these three values:  
#      enforcing - SELinux security policy is enforced.  
#      permissive - SELinux prints warnings instead of enforcing.  
#      disabled - No SELinux policy is loaded.  
SELINUX=enforcing  
# SELINUXTYPE= can take one of these two values:  
#      targeted - Targeted processes are protected,  
#      mls - Multi Level Security protection.  
SELINUXTYPE=targeted
```

Step 8    Restart the system and check the SELinux mode.

```
[root@Server ~]# getenforce  
Enforcing
```

Step 9    Enable the SELinux log service.

```
[root@Server ~]# systemctl enable --now auditd.service  
[root@Server ~]# systemctl status auditd.service  
● auditd.service - Security Auditing Service  
    Loaded: loaded (/usr/lib/systemd/system/auditd.service; enabled; vendor preset: enabled)  
    Active: active (running) since Tue 2023-05-16 13:30:09 UTC; 5s ago  
      Docs: man:auditd(8)  
            https://github.com/linux-audit/audit-documentation  
        Process: 2482 ExecStart=/sbin/auditd (code=exited, status=0/SUCCESS)  
        Process: 2486 ExecStartPost=/sbin/augenrules --load (code=exited, status=0/SUCCESS)  
      Main PID: 2483 (auditd)  
         Tasks: 2 (limit: 22471)  
        Memory: 856.0K  
       CGroup: /system.slice/auditd.service  
              └─2483 /sbin/auditd
```

### 3.2.2 Restricting SELinux Ports

Step 1    Ensure that the httpd service is running.

```
[root@Server ~]# systemctl start httpd.service  
[root@Server ~]# systemctl status httpd  
● httpd.service - The Apache HTTP Server  
    Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)  
    Active: active (running) since Tue 2023-05-16 13:33:19 UTC; 2min 5s ago  
      Docs: man:httpd.service(8)
```

Step 2    Check the context of the httpd process.

```
[root@Server ~]# ps -eZ | grep httpd  
system_u:system_r:httpd_t:s0      2524 ?      00:00:00 httpd  
system_u:system_r:httpd_t:s0      2525 ?      00:00:00 httpd  
system_u:system_r:httpd_t:s0      2526 ?      00:00:00 httpd  
system_u:system_r:httpd_t:s0      2527 ?      00:00:00 httpd  
system_u:system_r:httpd_t:s0      2528 ?      00:00:00 httpd
```

Step 3    Stop the httpd service and check its status.

```
[root@Server ~]# systemctl stop httpd  
[root@Server ~]# systemctl status httpd  
○ httpd.service - The Apache HTTP Server  
    Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)  
    Active: inactive (dead) since Tue 2023-05-16 13:40:46 UTC; 11s ago  
      Docs: man:httpd.service(8)
```

Step 4    Check the SELinux ports that allow for httpd listening.

```
[root@Server ~]# semanage port -l | grep -w http_port_t
http_port_t      tcp      80, 81, 443, 488, 8008, 8009, 8443, 9000
```

Step 5 Change the listening port of the httpd service to 82.

```
[root@Server ~]# vim /etc/httpd/conf/httpd.conf
Listen 192.168.0.49:82
```

Step 6 Start the httpd service again.

The httpd service fails to be started.

```
[root@Server ~]# systemctl start httpd
Job for httpd.service failed because the control process exited with error code.
See "systemctl status httpd.service" and "journalctl -xeu httpd.service" for details.
```

Step 7 Check SELinux logs.

Due to port restrictions, SELinux rejects httpd listening on port 82.

```
[root@Server ~]# ausearch -m avc -c httpd
-----
time-->Tue May 16 13:42:46 2023
type=AVC msg=audit(1684244566.444:21): avc: denied { name_bind } for pid=2726
comm="httpd" src=82 scontext=system_u:system_r:httpd_t:s0
tcontext=system_u:object_r:reserved_port_t:s0 tclass=tcp_socket permissive=0
```

Step 8 Configure SELinux and allow httpd to listen to port 82.

```
semanage port -a -t http_port_t -p tcp 82
```

Step 9 Restart the httpd service.

```
[root@Server ~]# systemctl start httpd
[root@Server ~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2023-05-16 13:50:38 UTC; 7s ago
     Docs: man:httpd.service(8)
```

Step 10 Verify the lab result.

```
[root@Server ~]# curl 192.168.0.49:82
<title>Test Page for the Apache HTTP Server on openEuler Linux</title>
```

### 3.2.3 Restricting SELinux Types

Step 1 Temporarily Change the SELinux mode to **permissive**.

```
[root@Server ~]# setenforce 0
[root@Server ~]# getenforce
```

### Permissive

#### Step 2 Create an httpd home page.

```
mkdir /home/mywebsite  
echo "hello,openEuler!" > /home/mywebsite/index.html  
chmod 755 /home/mywebsite
```

#### Step 3 Modify httpd configurations.

Modify the home directory of static resources and add the access permission.

```
[root@Server ~]# vim /etc/httpd/conf/httpd.conf  
#DocumentRoot "/var/www/html"  
DocumentRoot "/home/mywebsite"  
<Directory "/home/mywebsite">  
    AllowOverride None  
    # Allow open access:  
    Require all granted  
</Directory>
```

#### Step 4 Reload and verify configurations.

If the returned content meets the expectation, the modification on the home page is successful.

```
[root@Server ~]# systemctl reload httpd  
[root@Server ~]# curl 192.168.0.49:82  
hello,openEuler!
```

#### Step 5 Change the SELinux mode to **enforcing** and verify the access.

The returned content does not meet the expectation.

```
[root@Server ~]# setenforce 1  
[root@Server ~]# curl 192.168.0.49:82  
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">  
<html><head>  
<title>403 Forbidden</title>  
</head><body>  
<h1>Forbidden</h1>  
<p>You don't have permission to access this resource.</p>  
</body></html>
```

#### Step 6 Check SELinux logs.

SELinux denies the httpd service's access to static resources because the context type does not match.

```
[root@Server ~]# ausearch -m avc -c httpd  
----  
time-->Wed May 17 02:17:53 2023
```

```
type=AVC msg=audit(1684289873.780:191): avc: denied { getattr } for pid=1958 comm="httpd" path="/mywebsite/index.html" dev="sda2" ino=1048579 scontext=system_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:default_t:s0 tclass=file permissive=0
```

### Step 7 Change the type of **index.html**.

```
[root@Server ~]# semanage fcontext -a -t httpd_sys_content_t "/home/mywebsite(/.*)?"  
[root@Server ~]# restorecon -R -v /home/mywebsite  
Relabeled /mywebsite from unconfined_u:object_r:default_t:s0 to  
unconfined_u:object_r:httpd_sys_content_t:s0  
Relabeled /mywebsite/index.html from unconfined_u:object_r:default_t:s0 to  
unconfined_u:object_r:httpd_sys_content_t:s0
```

### Step 8 Change the Boolean value.

```
[root@Server ~]# setsebool -P httpd_enable_homedirs=on
```

### Step 9 Verify the lab result again.

The expected content is returned due to type consistency.

```
[root@Server ~]# curl 192.168.0.49:82  
hello,openEuler!
```

## 3.3 Quiz

- What are the differences between using the **chcon** and **semanage fcontext** commands to modify the SELinux context?

Answer: The SELinux context modified by **chcon** is not recorded in the file system. You can run **restorecon** to restore the SELinux context to the default context. By contrast, the SELinux context modified by **semanage fcontext** is recorded in the file system, and the default context is modified.

- Can SELinux replace firewalld to protect hosts?

Answer: SELinux can be used to guarantee data confidentiality and integrity and protect processes from untrusted input, but it cannot replace security software such as firewalld.

Huawei openEuler Certification Training

**HCIP-openEuler**

**System Monitoring**

**Lab Guide**

ISSUE: 1.0



HUAWEI TECHNOLOGIES CO., LTD

**Copyright © Huawei Technologies Co., Ltd. 2024. All rights reserved.**

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

### **Trademarks and Permissions**



HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

### **Notice**

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

## **Huawei Technologies Co., Ltd.**

Address:       Huawei Industrial Base Bantian, Longgang Shenzhen 518129  
                  People's Republic of China

Website:      <https://e.huawei.com>

## Huawei Certification System

Huawei Certification is an integral part of the company's Platform + Ecosystem strategy. It supports the development of ICT infrastructure that features Cloud-Pipe-Device synergy. Our certification is always evolving to reflect the latest trends in ICT development.

Huawei Certification consists of three categories: ICT Infrastructure Certification, Basic Software & Hardware Certification, and Cloud Platform & Services Certification, making it the most extensive technical certification program in the industry.

Huawei offers three levels of certification: Huawei Certified ICT Associate (HCIA), Huawei Certified ICT Professional (HCIP), and Huawei Certified ICT Expert (HCIE).

Our programs cover all ICT fields and follow the industry's trend of ICT convergence. With our leading talent development system and certification standards, we are committed to fostering new digital ICT talent and building a sound ICT talent ecosystem.

HCIP-openEuler is mainly for frontline engineers from Huawei and representative offices and readers who wish to learn openEuler O&M technologies. HCIP-openEuler certification covers common openEuler enterprise service management, openEuler HA cluster architecture, openEuler storage management, openEuler automated O&M, Linux shell scripts, openEuler system security hardening, and openEuler system monitoring.

Huawei certification helps you unlock opportunities to advance your career and take one more step towards the top of the industry.

## Huawei Career Certification

		Huawei Certified ICT Expert					
		Huawei Certified ICT Professional			Huawei Certified ICT Associate		
Cloud Platform & Services		Cloud Computing	Cloud Service	Big Data	GaussDB	IoT	
Basic Software & Hardware		openEuler	openGauss	HarmonyOS	AI	Kunpeng	
ICT Infrastructure	IT Infrastructure Domain	Storage	Collaboration	Intelligent Vision	Digital Power	MDC	
	Datacom Domain	Datacom		WLAN		Security	
	Optical Domain		Transmission		Access		
	Wireless Domain	5G			LTE		

# About This Document

---

## Overview

This document is an HCIP-openEuler certification training course and is intended for trainees who are going to take the HCIP-openEuler exam or readers who want to learn about system performance metrics, basic monitoring commands, and Zabbix on openEuler and other Linux distributions.

## Description

This lab guide consists of six labs, covering basic usage of system monitoring commands and Zabbix monitoring setup.

- Lab 1: CPU performance metric practice
- Lab 2: memory performance metric practice
- Lab 3: drive I/O performance metric practice
- Lab 4: network I/O performance metric practice
- Lab 5: **top** performance analysis practice
- Lab 6: Zabbix monitoring practice

## Background Knowledge Required

This course is for Huawei certification. To better understand this course, you need to:

- Have basic Linux knowledge. You are advised to complete HCIA-openEuler learning and pass the HCIA-openEuler certification exam.

## Lab Environment Preparation

### Checking Devices

Before starting the labs, each group of trainees should apply for ECSs on Huawei Cloud according to the following table.

Device Name	Specifications	Remarks
openEuler ECSs	2 vCPUs   4 GiB   s7.large.2	Prepare two ECSs.

# Contents

---

<b>About This Document .....</b>	<b>3</b>
Overview .....	3
Description .....	3
Background Knowledge Required .....	3
Lab Environment Preparation .....	3
<b>1 Basic System Monitoring Command Practice .....</b>	<b>1</b>
1.1 Introduction .....	1
1.1.1 About This Lab .....	1
1.1.2 Objectives .....	1
1.2 CPU Performance Metric Practice .....	1
1.2.1 Installing Lab Tools .....	1
1.2.2 Viewing CPU Usage .....	1
1.2.3 Viewing Context Switches .....	2
1.3 Memory Performance Metric Practice .....	3
1.3.1 Creating a Swap Partition .....	3
1.3.2 Checking System Memory .....	4
1.3.3 Checking Process Memory .....	5
1.4 Drive I/O Performance Metric Practice .....	5
1.4.1 Installing Lab Tools .....	5
1.4.2 Checking the Drive IOPS and Throughput .....	6
1.5 Network I/O Performance Metric Practice .....	6
1.5.1 Installing Lab Tools .....	6
1.5.2 Checking the Network Connection Status .....	7
1.5.3 Checking the Network Response Time .....	7
1.5.4 Checking the Network Bandwidth .....	7
1.6 <b>top</b> Performance Analysis Tool Practice .....	8
1.6.1 Interactive Operations on the Main Interface .....	8
1.7 Quiz .....	11
<b>2 Practice of Zabbix Monitoring .....</b>	<b>12</b>
2.1 Zabbix Installation and Deployment .....	12
2.1.1 Resource Preparations .....	12
2.1.2 Zabbix Configuration .....	14
2.1.3 Install Zabbix Agent .....	18
2.1.4 Basic Zabbix Operations .....	19



2.2 Quiz .....	27
----------------	----

# 1

# Basic System Monitoring Command Practice

---

## 1.1 Introduction

### 1.1.1 About This Lab

This lab describes how to use the basic commands for monitoring the CPU, memory, and I/O performance metrics on openEuler.

### 1.1.2 Objectives

- Master the usage of basic system monitoring commands.
- Understand the meanings of CPU, memory, and I/O performance metrics.

## 1.2 CPU Performance Metric Practice

### 1.2.1 Installing Lab Tools

Step 1    Install the system stress test tool stress.

```
[root@openEuler ~]# dnf config-manager --add-repo  
https://repo.oepkgs.net/openeuler/rpm/openEuler-22.03-LTS/extras/x86_64/  
[root@openEuler ~]# dnf -y install stress
```

Step 2    Install the thread stress test tool sysbench.

```
dnf -y install sysbench
```

Step 3    Install the system performance monitoring tool sysstat.

```
dnf -y install sysstat
```

### 1.2.2 Viewing CPU Usage

Step 1    Perform a CPU performance stress test.

```
[root@openEuler ~]# stress -c 2 -i 2 --timeout 300s &  
[1] 6042
```

stress: info: [6042] dispatching hogs: 2 cpu, 2 io, 0 vm, 0 hdd

### Step 2 View the CPU usage.

```
[root@openEuler ~]# mpstat -P ALL 5 2
Linux 5.10.0-60.18.0.50.oe2203.x86_64 (openEuler)           05/19/2023      _x86_64_      (2 CPU)

02:34:44 AM   CPU   %usr   %nice   %sys %iowait   %irq   %soft   %steal   %guest   %gnice   %idle
02:34:49 AM   all   51.13   0.00   3.38  45.29   0.20   0.00   0.00   0.00   0.00   0.00
02:34:49 AM     0   0.00   0.00   6.93  92.86   0.21   0.00   0.00   0.00   0.00   0.00
02:34:49 AM     1   99.80   0.00   0.00   0.00   0.20   0.00   0.00   0.00   0.00   0.00

02:34:49 AM   CPU   %usr   %nice   %sys %iowait   %irq   %soft   %steal   %guest   %gnice   %idle
02:34:54 AM   all   50.51   0.00   6.58  42.51   0.20   0.10   0.00   0.00   0.00   0.10
02:34:54 AM     0   79.48   0.00   1.01  19.11   0.20   0.20   0.00   0.00   0.00   0.00
02:34:54 AM     1   21.18   0.00  12.22  66.19   0.20   0.00   0.00   0.00   0.00   0.20

Average:   CPU   %usr   %nice   %sys %iowait   %irq   %soft   %steal   %guest   %gnice   %idle
Average:   all   50.81   0.00   4.99  43.89   0.20   0.05   0.00   0.00   0.00   0.05
Average:     0   40.60   0.00   3.91  55.19   0.21   0.10   0.00   0.00   0.00   0.00
Average:     1   60.85   0.00   6.05  32.80   0.20   0.00   0.00   0.00   0.00   0.10
```

### Step 3 Check the CPU usage of each process.

```
[root@openEuler ~]# ps aux --sort -%cpu
USER          PID %CPU %MEM      VSZ   RSS TTY      STAT START   TIME COMMAND
root        1285 47.5  0.0   3432   104 pts/0    R+   17:53  0:02 stress -c 2 -i 2
root        1282 47.3  0.0   3432   104 pts/0    R+   17:53  0:02 stress -c 2 -i 2
root        1283 47.3  0.0   3432   104 pts/0    R+   17:53  0:02 stress -c 2 -i 2
root        1284 47.3  0.0   3432   104 pts/0    R+   17:53  0:02 stress -c 2 -i 2
```

### Step 4 Check the average system load.

[root@openEuler ~]# uptime  
02:39:52 up 1:45, 2 users, load average: 0.02, 0.47, 0.44

## 1.2.3 Viewing Context Switches

### Step 1 Perform a context switch stress test.

```
[root@openEuler ~]# sysbench --threads=4 --time=300 threads run &
[1] 6086
sysbench 1.0.20 (using system LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 4
...
```

### Step 2 Check system context switches.

```
[root@openEuler ~]# vmstat 3 5
procs -----memory----- swap-- -----io---- -system-- -----cpu-----
r b    swpd    free    buff  cache    si    so    bi    bo   in    cs us sy id wa st
5 0      0 3210240  16900 336800    0    0    14   1313  123  521  8  3 88  1  0
4 0      0 3210240  16900 336800    0    0     0    0 2014 3121943 23 78  0  0  0
4 0      0 3210240  16900 336800    0    0     0    0 2015 3123714 23 77  0  0  0
4 0      0 3210240  16900 336800    0    0     0    0 2012 3120485 23 78  0  0  0
4 0      0 3210240  16900 336800    0    0     0    0 2013 3122594 22 78  0  0  0
```

### Step 3 Check thread context switches.

In the multi-threaded scenario, the system load increases as the number of CPU context switches per second reaches 3.2 million, accompanied by an increase in both voluntary and involuntary task switches.

```
[root@openEuler ~]# pidstat -w 5 2 -t | grep sysbench
03:04:27 AM    0      1969      1970    18.60 780629.80  (sysbench)_sysbench
03:04:27 AM    0      -      1971     0.60 779795.80  |_sysbench
03:04:27 AM    0      -      1972     0.20 779818.40  |_sysbench
03:04:27 AM    0      -      1973    87.00 780577.40  |_sysbench
03:04:32 AM    0      1969      1970   336.00 779162.00  (sysbench)_sysbench
03:04:32 AM    0      -      1971    11.00 780888.60  |_sysbench
03:04:32 AM    0      -      1972   168.20 779154.40  |_sysbench
03:04:32 AM    0      -      1973     0.40 780773.80  |_sysbench
Average:      0      1969      1970   177.30 779895.90  (sysbench)_sysbench
Average:      0      -      1971     5.80 780342.20  |_sysbench
Average:      0      -      1972    84.20 779486.40  |_sysbench
Average:      0      -      1973    43.70 780675.60  |_sysbench
```

### Step 4 Check the average system load.

```
[root@openEuler ~]# uptime
03:14:46 up 2:20, 2 users, load average: 4.90, 3.12, 1.89
```

## 1.3 Memory Performance Metric Practice

### 1.3.1 Creating a Swap Partition

#### Step 1 Generate a 2 GiB data file.

```
[root@openEuler ~]# dd if=/dev/zero of=/mnt/swap bs=1M count=2048
2048+0 records in
2048+0 records out
2147483648 bytes (2.1 GB, 2.0 GiB) copied, 11.4539 s, 187 MB/s
```

#### Step 2 Format the file and check file information.

```
[root@openEuler ~]# mkswap /mnt/swap
mkswap: /mnt/swap: insecure permissions 0644, fix with: chmod 0600 /mnt/swap
Setting up swapspace version 1, size = 2 GiB (2147479552 bytes)
no label, UUID=8e4e45fb-253a-45c8-a530-f13ed8a606c3
[root@openEuler ~]# file /mnt/swap
/mnt/swap: Linux swap file, 4k page size, little endian, version 1, size 524287 pages, 0 bad pages, no
label, UUID=8e4e45fb-253a-45c8-a530-f13ed8a606c3
```

#### Step 3 Modify the permissions on the partition file and activate the swap partition.

```
chmod 0600 /mnt/swap
swapon /mnt/swap
```

#### Step 4 Check the swap partition size.

```
[root@openEuler ~]# swapon -s
Filename                                Type      Size    Used   Priority
/mnt/swap                                file     2097148     0      -2
[root@openEuler ~]# free -h
              total        used         free       shared  buff/cache   available
Mem:       3.5Gi       67Mi       1.1Gi      16Mi       2.3Gi       3.1Gi
Swap:      2.0Gi        0B        2.0Gi
```

### 1.3.2 Checking System Memory

#### Step 1 Perform a physical memory performance stress test.

```
[root@openEuler ~]# stress --vm 2 --vm-bytes 300M --vm-keep --timeout 300s &
[1] 6091
stress: info: [1247] dispatching hogs: 0 cpu, 0 io, 2 vm, 0 hdd
```

#### Step 2 Check the physical memory.

```
[root@openEuler ~]# free -h
              total        used         free       shared  buff/cache   available
Mem:       3.5Gi       671Mi      560Mi      16Mi       2.3Gi       2.6Gi
Swap:      2.0Gi        0B        2.0Gi
```

#### Step 3 Check the swappiness and priority of the swap space.

**swappiness** controls memory reclamation: the higher the value, the more actively the kernel will use the swap space.

**priority** indicates the priority of the swap space: the higher the value, the higher the priority, and the more likely the kernel will use it.

```
[root@openEuler ~]# cat /proc/sys/vm/swappiness
60
[root@openEuler ~]# swapon -s
Filename                                Type      Size    Used   Priority
/mnt/swap                                file     2097148   39852      -2
```

#### Step 4 Perform a swap space memory stress test.

```
[root@openEuler ~]# stress --vm 7 --vm-bytes 500M --vm-hang 5 --timeout 300s &
[1] 6091
stress: info: [1361] dispatching hogs: 0 cpu, 0 io, 7 vm, 0 hdd
```

#### Step 5 Check the swap space memory.

```
[root@openEuler ~]# free -h
              total        used         free       shared  buff/cache   available
Mem:       3.5Gi       3.3Gi      105Mi      8.0Mi       47Mi       9.0Mi
Swap:      2.0Gi       1.1Gi      941Mi
```

## Step 6 Check the system virtual memory.

```
[root@openEuler ~]# vmstat -a -S M 3 5
procs -----memory----- swap-----io----system-----cpu-----
 r b swpd   free  inact active   si   so   bi   bo   in   cs us sy id wa st
 4 4    438    126  3329      7    1    1  934 1984 201   89 14  1 82  3  0
 1 5    652    114  3348      7   50   71 51080 72876 2454 3905  0  5 2 92  0
 1 5    818     87  3382      9   60   55 61960 56975 2558 3823  0  7 0 93  0
 1 5   1060    105  3364      9   34   86 34880 88061 2437 4394  0  5 1 94  0
 2 4   1105    108  3351      9   39   84 40243 85855 2534 4024  0  6 4 90  0
```

### 1.3.3 Checking Process Memory

#### Step 1 Perform a process memory stress test.

```
[root@openEuler ~]# sysbench --threads=4 --time=300 threads run --timeout 300s &
[1] 6091
sysbench 1.0.20 (using system LuaJIT 2.1.0-beta3)
```

Running the test with following options:

Number of threads: 4

...

#### Step 2 Check the process memory.

- On the monitoring interface of **top**, press the following keys in sequence: **h > h > 3 > Enter > J > H > e**

Press the above keys to change the functions and format of the **top** main interface and check the process memory details.

```
top - 15:48:02 up 1:52, 2 users, load average: 3.97, 3.23, 1.82
Threads: 89 total, 5 running, 84 sleeping, 0 stopped, 0 zombie
%Cpu(s): 21.5 us, 78.2 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.3 hi, 0.0 si, 0.0 st
MiB Mem : 3546.2 total, 3454.1 free, 52.1 used, 40.1 buff/cache
MiB Swap: 2048.0 total, 2009.8 free, 38.2 used. 3358.0 avail Mem
```

PID	%MEM	VIRT	RES	CODE	DATA	SHR	nMaj	nDRT	%CPU	COMMAND
1484	0.2	14.9m	5.5m	0.1m	3.9m	4.8m	0	0	0.0	sysbench
1485	0.2	14.9m	5.5m	0.1m	3.9m	4.8m	0	0	50.0	sysbench
1486	0.2	14.9m	5.5m	0.1m	3.9m	4.8m	0	0	50.0	sysbench
1487	0.2	14.9m	5.5m	0.1m	3.9m	4.8m	0	0	50.0	sysbench
1488	0.2	14.9m	5.5m	0.1m	3.9m	4.8m	0	0	50.0	sysbench
1481	0.1	9.3m	3.8m	0.1m	1.2m	3.2m	7	0	0.0	top

## 1.4 Drive I/O Performance Metric Practice

### 1.4.1 Installing Lab Tools

#### Step 1 Install the drive I/O test tool.

```
dnf -y install fio
```

Step 2 Create a directory for stress test data.

```
mkdir /data
```

### 1.4.2 Checking the Drive IOPS and Throughput

Step 1 Perform a stress test on drive I/O performance.

```
[root@openEuler ~]# fio -filename=/data/test.file -direct=1 -iodepth 1 -thread -rw=randrw -rwmixread=70 -ioengine=psync -bs=16k -size=2G -numjobs=10 -runtime=60 -group_reporting -name=test_r_w  
test_r_w: (g=0): rw=randrw, bs=(R) 16.0KiB-16.0KiB, (W) 16.0KiB-16.0KiB, (T) 16.0KiB-16.0KiB,  
ioengine=psync, iodepth=1  
...
```

Step 2 Check the drive IOPS and throughput.

Start another terminal. You can see that the IOPS of drive **sda** is 7,705 and the throughput is 120 Mbit/s.

```
[root@openEuler ~]# iostat -dh 3 3  
Linux 5.10.0-60.18.0.50.oe2203.x86_64 (openEuler)          05/19/2023      _x86_64_      (2 CPU)  
  
    tps   kB_read/s   kB_wrtn/s   kB_dscd/s   kB_read   kB_wrtn   kB_dscd Device  
 276.02     5.8M       7.0M       0.0k    62.8G    76.1G     0.0k sda  
  
    tps   kB_read/s   kB_wrtn/s   kB_dscd/s   kB_read   kB_wrtn   kB_dscd Device  
7705.67    83.9M      36.5M       0.0k   251.8M   109.4M     0.0k sda  
  
    tps   kB_read/s   kB_wrtn/s   kB_dscd/s   kB_read   kB_wrtn   kB_dscd Device  
7573.33    83.0M      35.3M       0.0k   249.0M   106.0M     0.0k sda
```

Step 3 Check the IOPS and throughput of block device **sda**.

```
[root@openEuler ~]# sar -dh 3 3  
Linux 5.10.0-60.18.0.50.oe2203.x86_64 (openEuler)          05/19/2023      _x86_64_      (2 CPU)  
  
05:03:17 PM      tps    rkB/s    wkB/s    dkB/s    areq-sz    aqu-sz    await    %util DEV  
05:03:20 PM    7680.33   83.6M   36.4M     0.0k    16.0k     9.97    1.30   100.0% sda  
05:03:23 PM    7680.33   83.9M   36.1M     0.0k    16.0k     9.99    1.30   100.0% sda  
05:03:26 PM    7680.00   83.9M   36.1M     0.0k    16.0k     9.95    1.30   100.0% sda  
Average:     7680.22   83.8M   36.2M     0.0k    16.0k     9.97    1.30   100.0% sda
```

## 1.5 Network I/O Performance Metric Practice

### 1.5.1 Installing Lab Tools

Step 1 Install the network performance test tool iperf.

```
dnf -y install iperf
```

## 1.5.2 Checking the Network Connection Status

Step 1 Check the network connection status.

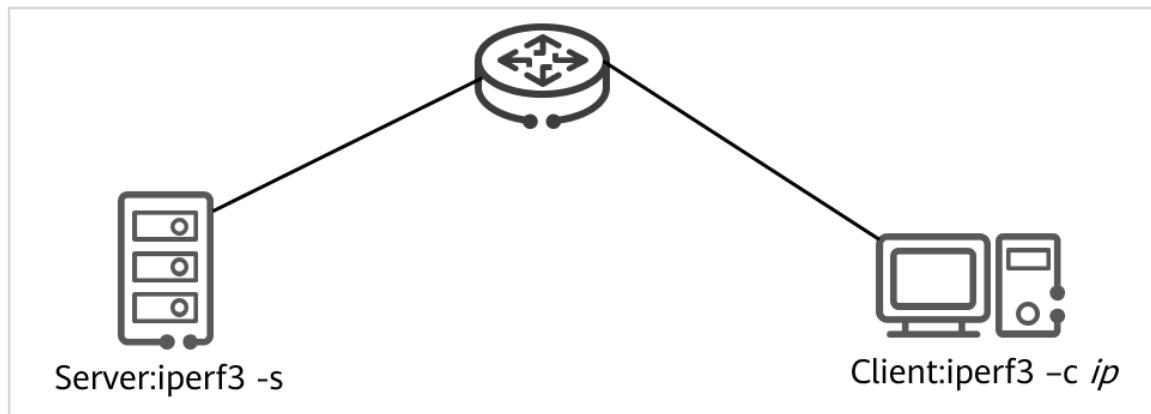
```
[root@openEuler ~]# netstat -antulp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address        State      PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*              LISTEN     494/sshd: /usr/sbin
tcp        0    276 192.168.0.48:22         119.3.119.19:27308    ESTABLISHED 1717/sshd: root [pr
tcp        0      0 192.168.0.48:22         119.3.119.19:27309    ESTABLISHED 1862/sshd: root [pr
tcp        0      0 192.168.0.48:22         119.3.119.19:27305    ESTABLISHED 1250/sshd: root [pr
tcp        0      0 192.168.0.48:22         119.3.119.19:27304    ESTABLISHED 896/sshd: root [pri]
```

## 1.5.3 Checking the Network Response Time

```
[root@openEuler ~]# ping www.huaweicloud.com -c 5
PING koa8myv3.sslego-dk.tcloudscdn.com (36.249.84.38) 56(84) bytes of data.
64 bytes from 36.249.84.38 (36.249.84.38): icmp_seq=1 ttl=44 time=28.9 ms
64 bytes from 36.249.84.38 (36.249.84.38): icmp_seq=2 ttl=44 time=28.8 ms
64 bytes from 36.249.84.38 (36.249.84.38): icmp_seq=3 ttl=44 time=28.7 ms
64 bytes from 36.249.84.38 (36.249.84.38): icmp_seq=4 ttl=44 time=28.8 ms
64 bytes from 36.249.84.38 (36.249.84.38): icmp_seq=5 ttl=44 time=28.8 ms

--- koa8myv3.sslego-dk.tcloudscdn.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 28.736/28.781/28.881/0.051 ms
```

## 1.5.4 Checking the Network Bandwidth



Step 1 Start the service on an ECS.

Use either ECS on the intranet as the server.

```
[root@Server ~]# iperf3 -s
-----
[  ] Server listening on 5201 (test #1)
-----
```

Step 2 Test the intranet bandwidth on the client.

You can see that the intranet bandwidth is about 4 Gbit/s.

```
[root@openEuler ~]# iperf3 -c 192.168.0.49 -t 20 -i 5
Connecting to host 192.168.0.49, port 5201
[ 5] local 192.168.0.48 port 44152 connected to 192.168.0.49 port 5201
[ ID] Interval Transfer Bitrate Retr Cwnd
[ 5] 0.00-5.00 sec 2.89 GBytes 4.97 Gbits/sec 35541 105 KBytes
[ 5] 5.00-10.00 sec 2.34 GBytes 4.02 Gbits/sec 48369 304 KBytes
[ 5] 10.00-15.00 sec 2.34 GBytes 4.02 Gbits/sec 42416 116 KBytes
[ 5] 15.00-20.00 sec 2.34 GBytes 4.02 Gbits/sec 45856 225 KBytes
- - - - -
[ ID] Interval Transfer Bitrate Retr
[ 5] 0.00-20.00 sec 9.91 GBytes 4.26 Gbits/sec 172182
[ 5] 0.00-20.00 sec 9.91 GBytes 4.25 Gbits/sec
                                         sender
                                         receiver

iperf Done.
```

### Step 3 Check the throughput of the network interface.

The throughput of the **ens3** interface is about 500,000 kilobytes per second.

```
[root@openEuler ~]# sar -n DEV 2 1
Linux 5.10.0-60.18.0.50.oe2203.x86_64 (openEuler)          05/20/2023      _x86_64_      (2 CPU)

09:16:18 PM     IFACE    rxpck/s    txpck/s    rxkB/s    txkB/s    rxcmp/s    txcmp/s    rxmcst/s    %ifutil
09:16:20 PM       lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
09:16:20 PM     ens3   25018.00  20775.00  1691.78 513374.11      0.00      0.00      0.00      0.00

Average:     IFACE    rxpck/s    txpck/s    rxkB/s    txkB/s    rxcmp/s    txcmp/s    rxmcst/s    %ifutil
Average:       lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
Average:     ens3   25018.00  20775.00  1691.78 513374.11      0.00      0.00      0.00      0.00
```

## 1.6 top Performance Analysis Practice

The **top** command is a commonly used Linux performance analysis tool. It can display the resource usage of each process in real time, similar to Task Manager on Windows. The real-time system status view provides a summary of system information and a list of processes or threads currently managed by the Linux kernel. The system summary information and the type, order, and unit of process resources can be customized and stored permanently.

### 1.6.1 Interactive Operations on the Main Interface

#### Step 1 View the system information on the main interface of **top**.

```
# top
```

The following information is displayed from top to bottom: system time and load, task status, CPU usage, memory and virtual memory, and processes under the fields.

```
top - 18:16:14 up 8:40, 2 users, load average: 0.00, 0.00, 0.00
Tasks: 102 total, 1 running, 101 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.2 us, 0.2 sy, 0.0 ni, 99.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 7220.3 total, 6568.1 free, 216.1 used, 436.2 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 6524.3 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
1031 root 20 0 2281700 29068 18920 S 0.3 0.4 0:37.30 hostguard
  1 root 20 0 103200 11508 8808 S 0.0 0.2 0:01.31 systemd
  2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthreadd
  3 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_gp
  4 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_par_gp
```

### Step 2 Move the cursor.

Press the arrow keys to scroll and view fields and processes in the task area.

### Step 3 Customize the main interface.

Key	Function	Key	Function
j	Aligns texts to the left or right.	B	Toggles the bold text mode.
J	Aligns numbers to the left or right.	z	Toggles between the multicolor and monochrome mode.

```
top - 10:05:41 up 40 min, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 99 total, 1 running, 98 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 7220.3 total, 6588.3 free, 219.9 used, 412.2 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 6521.9 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
1 root 20 0 103204 11464 8764 S 0.0 0.2 0:01.06 systemd
2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthreadd
3 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_gp
4 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_par_gp
6 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/0:0H-kblockd
```

In this lab, numbers are left-aligned, texts are right-aligned, the multicolor mode is used, and keywords are in bold.

### Step 4 Adjust the information displayed in the summary area.

Press keys for interactive commands to display or hide information in the summary area or adjust information display.

Key	Function
l	Displays or hides the average system load.
t	Displays or hides task and CPU statistics, and adjusts the display mode of CPU statistics.

Key	Function
m	Displays or hides memory information, and adjusts the display mode of memory information.
1	Toggles between per-CPU usage and average CPU usage.

```
top - 11:29:55 up 2:04, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 98 total, 1 running, 97 sleeping, 0 stopped, 0 zombie
%CPU0 : 0.0/0.3 0[ ]
%CPU1 : 0.0/0.0 0[ ]
%CPU2 : 0.3/0.0 0[ ]
%CPU3 : 0.0/0.0 0[ ]
MiB Mem : 9.7/7220.3 [|||||||||]
MiB Swap: 0.0/0.0 [ ]
```

PID	USER	PR	NI	VIRT	RES	SHR	S %CPU	%MEM	TIME+	COMMAND
921	root	20	0	138912	10860	9284	S 0.3	0.1	0:01.17	hostwatch
936	root	20	0	2281700	28332	18900	S 0.3	0.4	0:09.06	hostguard
1	root	20	0	103204	11464	8764	S 0.0	0.2	0:01.10	systemd
2	root	20	0	0	0	0	S 0.0	0.0	0:00.00	kthreadd
3	root	0	-20	0	0	0	I 0.0	0.0	0:00.00	rcu_gp

In this lab, the usage of each CPU is displayed, and the display mode of CPU and memory statistics in the summary area is modified.

### Step 5 Sort and modify fields in the task area.

Press **f** to go to the field management page. You can check the meaning of each field, select required fields, and sort them.

Key	Function	Key	Function
d	Shows or hides a field.	s	Specifies the field by which processes will be sorted.
Right	Selects a field.	Left	Confirms the location of a field.
Up/Down	Adjusts the location of a field.	q	Exits the field management page.

```
top - 11:00:58 up 1:35, 1 user, load average: 0.00, 0.01, 0.00
Tasks: 99 total, 1 running, 98 sleeping, 0 stopped, 0 zombie
%CPU(s): 0.0/0.6 1[ ]
MiB Mem : 9.7/7220.3 [|||||||||]
MiB Swap: 0.0/0.0 [ ]
```

PPID	PID	USER	PR	NI	VIRT	RES	SHR	S %CPU	%MEM	COMMAND	ENVIRON
0	1	root	20	0	103204	11464	8764	S 0.0	0.2	systemd	TERM=vt220 BOOT_IMAG
0	2	root	20	0	0	0	0	S 0.0	0.0	kthreadd	-
2	3	root	0	-20	0	0	0	I 0.0	0.0	rcu_gp	-
2	4	root	0	-20	0	0	0	I 0.0	0.0	rcu_par_gp	-
2	6	root	0	-20	0	0	0	I 0.0	0.0	kworker/0:0H-kblockd	-

In this lab, processes are sorted by the **PID** field, the **TIME+** field is hidden, the **PPID** and **ENVIRON** fields are added for display, and the **PPID** field is moved to the first column.

(1) The content under the **ENVIRON** field is not fully displayed. You can press the right arrow key to scroll the view for the complete content.

(2) By default, processes in the task area are sorted in descending order based on the selected field. You can press **R** to reverse the sorting order.

Step 6 Adjust the information displayed in the task area.

Key	Function	Key	Function
R	Sorts process in ascending or descending order.	n	Sets the maximum number of processes for display.
c	Displays process names or startup commands.	x	Toggles highlighting of the sorting column.
v	Toggles the tree view.	y	Toggles highlighting of running tasks.
< / >	Switches the sorting field.	H	Shows or hides thread information.

```

top - 11:52:58 up 2:27, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 99 total, 1 running, 98 sleeping, 0 stopped, 0 zombie
%Cpu0 : 0.0/0.0 0[          ]
%Cpu1 : 0.0/0.0 0[          ]
%Cpu2 : 0.0/0.0 0[          ]
%Cpu3 : 0.0/0.0 0[          ]
MiB Mem : 9.7/7220.3 [|||||||||] 9.7%
MiB Swap: 0.0/0.0 [          ] 0.0%
]

PID   PID   USER   PR NI VIRT   RES   SHR S %CPU COMMAND           ENVIRON
1    2188  root   20  0 13448  7192  6344 S 0.0  /usr/sbin/sshd -D
2188 2735  root   20  0 15100  8952  7720 S 0.0  ` sshd: root [priv]
2735 2737  root   20  0 15100  5140  3988 S 0.0  ` sshd: root@pts/1
2737 2738  root   20  0 214160 3932  3344 S 0.0  ` bash
2738 3190  root   20  0 216664 4060  3356 R 0.0  ` top
                                         LANG=en_US.UTF-8 PATH=/usr/local/sbin:+
                                         + PATH+USER=root LOGNAME=root HOME=/root PATH+
                                         + SHELL=/bin/bash HISTCONTROL= HISTSIZE=+

```

In this lab, a maximum of five processes are displayed, the tree view is enabled to visualize parent-child process relationships, and the sorting field and running processes are highlighted.

Step 7 Save the configuration permanently.

Press **W** to save the modified configuration to a file. If the following information is displayed, the configuration is successfully saved:

Wrote configuration to '/root/.config/procps/toprc'

## 1.7 Quiz

- How do you adjust the frequency at which the system uses the swap space?

Answer: Adjust the swappiness and priority of the swap space.

- What is the difference between **buff** and **cache** in memory performance metrics?

Answer: **buff** is used to reduce the number of I/O responses and handle excessive resource accesses. **cache** is a compromise strategy used to handle speed mismatches and accelerate accesses.

# 2 Practice of Zabbix Monitoring

## 2.1 Zabbix Installation and Deployment

### 2.1.1 Resource Preparations

Zabbix can use Nginx's resources deployed in previous labs. Alternatively, you can create a new VM to install and deploy Zabbix. This lab uses a new VM as an example. If you want to reuse Nginx's resources, skip LNMP setup and just modify the Nginx configuration file.

#### Step 1 Configure LNMP.

For details, see previous lab guides.

1. Install Nginx.

2 Install and configure PHP.

Install PHP and related dependencies.

```
yum install -y php-cli php-fpm php-gd php-mbstring php-bcmath php-xml php-ldap php-mysqlnd
```

3. Modify PHP configurations based on Zabbix requirements.

```
sed -i 's/post_max_size = 8M/post_max_size = 16M/g' /etc/php.ini  
sed -i 's/max_execution_time = 30/max_execution_time = 300/g' /etc/php.ini  
sed -i 's/max_input_time = 60/max_input_time = 300/g' /etc/php.ini
```

4. Modify the Nginx configuration file.

```
server {  
    listen 0.0.0.0:80;  
    root /data/zabbix;  
    index index.php;  
    location ~ \.php$ {  
        fastcgi_pass 127.0.0.1:9000;  
        fastcgi_index index.php;  
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;  
        include fastcgi_params;  
    }  
}
```

#### Step 2 Download and compile Zabbix.

Download the Zabbix source package.

```
 wget https://cdn.zabbix.com/zabbix/sources/oldstable/6.2/zabbix-6.2.8.tar.gz
```

After the download, run the **tar** command to decompress the package.

```
 tar -zxvf zabbix-6.2.8.tar.gz
```

Create the user and user group required by Zabbix.

```
 groupadd --system zabbix  
 useradd --system -g zabbix -d /usr/lib/zabbix -s /sbin/nologin -c "Zabbix Monitoring System" zabbix
```

Before compiling Zabbix, install required dependencies.

```
 yum install -y mysql-devel pcre-devel openssl-devel zlib-devel libxml2-devel net-snmp-devel net-snmp libssh2-devel OpenIPMI-devel libevent-devel openldap-devel libcurl-devel
```

Go to the decompressed directory and configure the Zabbix compilation file.

```
 mkdir /etc/zabbix  
 ./configure --sysconfdir=/etc/zabbix --enable-server --enable-agent --with-mysql --with-ssh2 --with-zlib --with-libpthread --with-libevent --with-libpcre --with-net-snmp --with-libcurl --with-libxml2 --with-openipmi --openssl --with-ldap
```

The following figure shows the information returned after the configuration.

```
 Enable agent 2:          no  
  
 Enable web service:      no  
  
 Enable Java gateway:     no  
  
 LDAP support:            no  
 IPv6 support:            no  
  
 ****  
 *           Now run 'make install'          *  
 *  
 *           Thank you for using Zabbix!      *  
 *           <http://www.zabbix.com>          *  
 ****
```

After the compilation is complete, install Zabbix.

```
 make install
```

The following figure shows the information returned after the compilation and installation.

```
make[2]: Leaving directory '/root/zabbix-6.0.14/man'
make[1]: Leaving directory '/root/zabbix-6.0.14/man'
Making install in misc
make[1]: Entering directory '/root/zabbix-6.0.14/misc'
make[2]: Entering directory '/root/zabbix-6.0.14/misc'
make[2]: Nothing to be done for 'install-exec-am'.
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/root/zabbix-6.0.14/misc'
make[1]: Leaving directory '/root/zabbix-6.0.14/misc'
make[1]: Entering directory '/root/zabbix-6.0.14'
make[2]: Entering directory '/root/zabbix-6.0.14'
make[2]: Nothing to be done for 'install-exec-am'.
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/root/zabbix-6.0.14'
make[1]: Leaving directory '/root/zabbix-6.0.14'
```

### Step 3 Configure a database.

The MySQL database is used in this lab, which shares the same VM with previous labs. You only need to create a database and user for Zabbix on the original database server.

```
create database zabbix charset utf8 collate utf8_bin;
create user 'zabbix'@'%' identified by 'Huawei@123';
grant all on zabbix.* to 'zabbix'@'%';
flush privileges;
```

After the preceding configuration is complete, copy the files to be imported from Zabbix to MySQL.

```
scp -r root@192.168.1.4:/root/zabbix-6.0.14/database/mysql .
```

Strictly follow the sequence as below; otherwise, an error will be reported.

```
mysql -uroot -pHuawei@123 zabbix < schema.sql
mysql -uroot -pHuawei@123 zabbix < images.sql
mysql -uroot -pHuawei@123 zabbix < data.sql
mysql -uroot -pHuawei@123 zabbix < double.sql
mysql -uroot -pHuawei@123 zabbix < history_pk_prepare.sql
```

So far, all preparations are complete.

## 2.1.2 Zabbix Configuration

### Step 1 Connect the Zabbix server to MySQL and start the Zabbix server.

Modify the Zabbix configuration file **/etc/zabbix/zabbix\_server.conf** and change the value of **DBHost** to the MySQL address (line 87).

```
# Mandatory: no  
# Default:  
DBHost=192.168.1.8
```

Change the value of **DBPassword** to the password specified during user creation (line 123).

```
# Mandatory: no  
# Default:  
DBPassword=Huawei@123
```

Specify **DBPort** to 3306 (line 140).

```
138 # Range: 1024-65535  
139 # Default:  
140 DBPort=3306
```

Uncomment **ListenPort** (line 12).

```
# Range: 1024-32767  
# Default:  
ListenPort=10051
```

Save the settings and exit. Run the **zabbix\_server -c /etc/zabbix/zabbix\_server.conf** command to start the service. If the service is started properly, the related port is listened on, as shown in the following figure.

```
[root@Zabbix zabbix-6.0.14]# ss -lnptu | grep 10051  
tcp  LISTEN  0      4096  0.0.0.0:10051  0.0.0.0:* users:(("zabbix server", pid=26600, fd=9), ("zabbix_server", pid=26599, fd=9), ("zabbix_server", pid=26598, fd=9), ("zabbix_server", pid=26594, fd=9), ("zabbix_server", pid=26593, fd=9), ("zabbix_server", pid=26592, fd=9), ("zabbix_server", pid=26588, fd=9), ("zabbix_server", pid=26587, fd=9), ("zabbix_server", pid=26581, fd=9), ("zabbix_server", pid=26577, fd=9), ("zabbix_server", pid=26576, fd=9), ("zabbix_server", pid=26575, fd=9), ("zabbix_server", pid=26571, fd=9), ("zabbix_server", pid=26570, fd=9), ("zabbix_server", pid=26569, fd=9), ("zabbix_server", pid=26565, fd=9), ("zabbix_server", pid=26564, fd=9), ("zabbix_server", pid=26559, fd=9), ("zabbix_server", pid=26558, fd=9), ("zabbix_server", pid=26557, fd=9))
```

## Step 2 Deploy the web interface.

Copy all files in the **ui** directory of the source package to the specified **root** directory of the Nginx service.

```
mkdir -p /data/zabbix/  
cp -r /root/zabbix-6.2.8/ui/* /data/zabbix/
```

Enter the Zabbix IP address in the address box of the browser to access Zabbix and deploy the web interface. The following figure shows its home page.



Click **Next** twice. On the database configuration page, complete the configuration as planned.

### Configure DB connection

Please create database manually, and set the configuration parameters for connection to this database. Press "Next step" button when done.

Database type	MySQL
Database host	192.168.1.8
Database port	0 0 - use default port
Database name	zabbix
Store credentials in	Plain text (selected)
User	zabbix
Password	*****
Database TLS encryption	<input checked="" type="checkbox"/>
Verify database certificate	<input type="checkbox"/>

At the bottom right are 'Back' and 'Next step' buttons.

After the configuration, click **Next step**. On the displayed page, set the parameters as shown in the following figure.

## Settings

Zabbix server name

Default time zone

Default theme

After the configuration, click **Next step**. On the displayed page, check whether the configuration is correct. If the configuration is correct, click **Next step**.

## Pre-installation summary

Please check configuration parameters. If all is correct, press "Next step" button, or "Back" button to change configuration parameters.

Database type MySQL

Database server 

Database port default

Database name zabbix

Database user zabbix

Database password \*\*\*\*\*

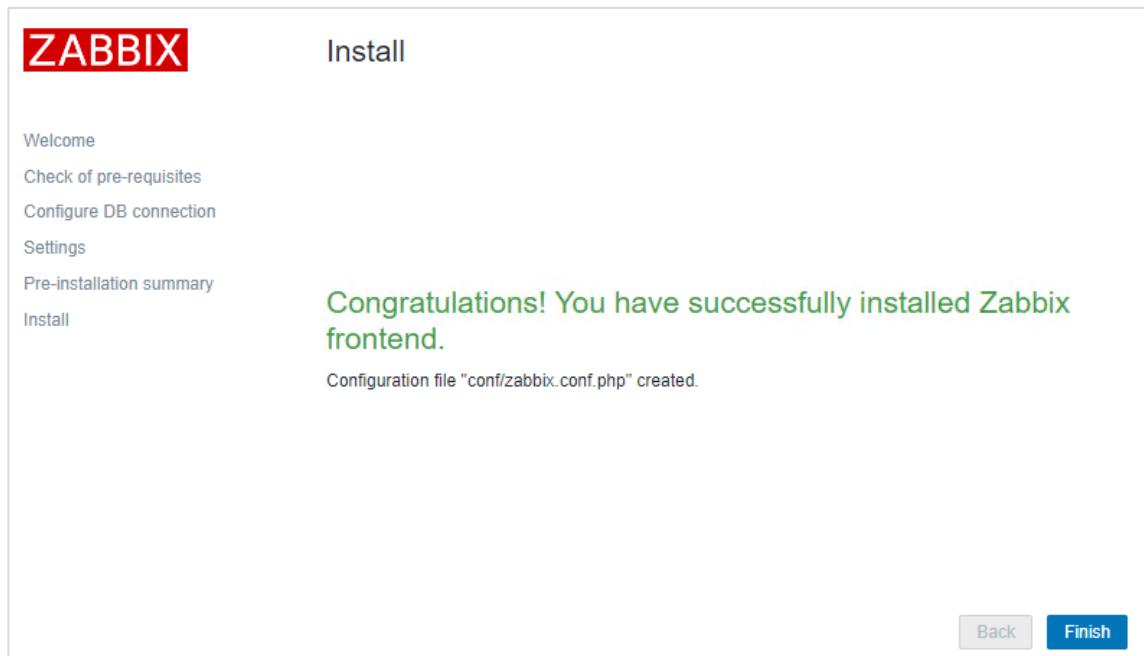
Database TLS encryption true

Zabbix server name Zabbix

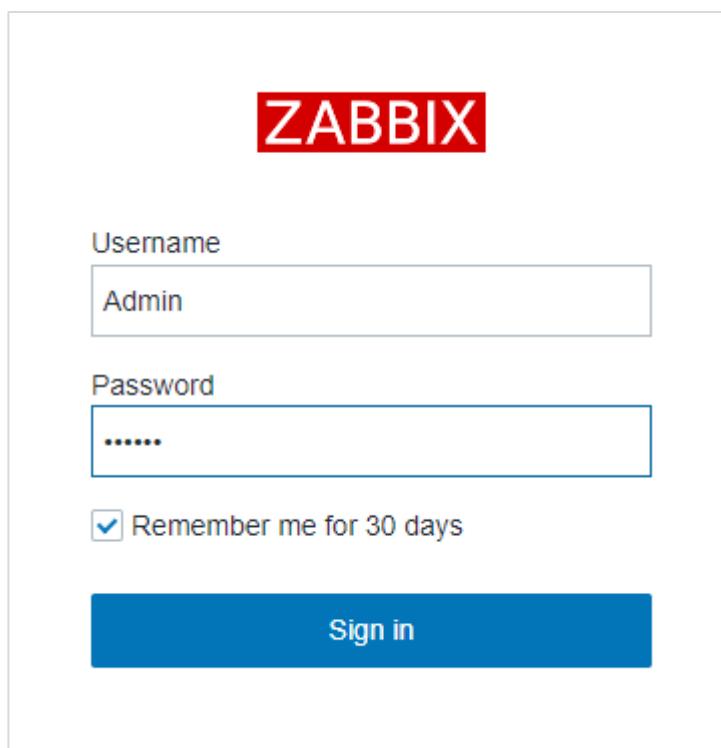
[Back](#)

[Next step](#)

After the configuration, the system displays a message indicating that the installation is complete, as shown in the following figure.



Click **Finish**. On the Zabbix web interface, enter the default username (Admin) and password (zabbix), and click **Sign In**.



## 2.1.3 Install Zabbix Agent

Step 1 Download and install the official Yum source file of Zabbix.

```
rpm -Uvh https://repo.zabbix.com/zabbix/6.2/rhel/8/x86\_64/zabbix-release-6.2-3.el8.noarch.rpm
```

Step 2 Run the **yum** command to install zabbix-agent2.

```
yum install -y zabbix-agent2
```

Step 3 Configure and start the agent.

Modify the agent configuration file and specify the IP address of Zabbix server.

```
sed -i 's/Server=127.0.0.1/Server=192.168.1.4/g' /etc/zabbix/zabbix_agent2.conf
```

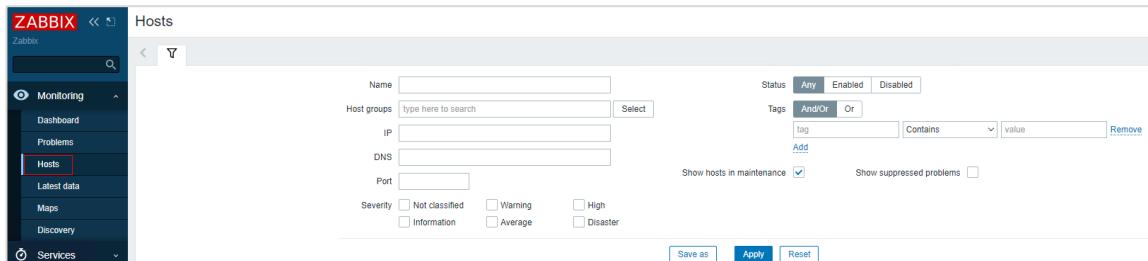
After the modification, start the agent.

```
systemctl start zabbix-agent2
```

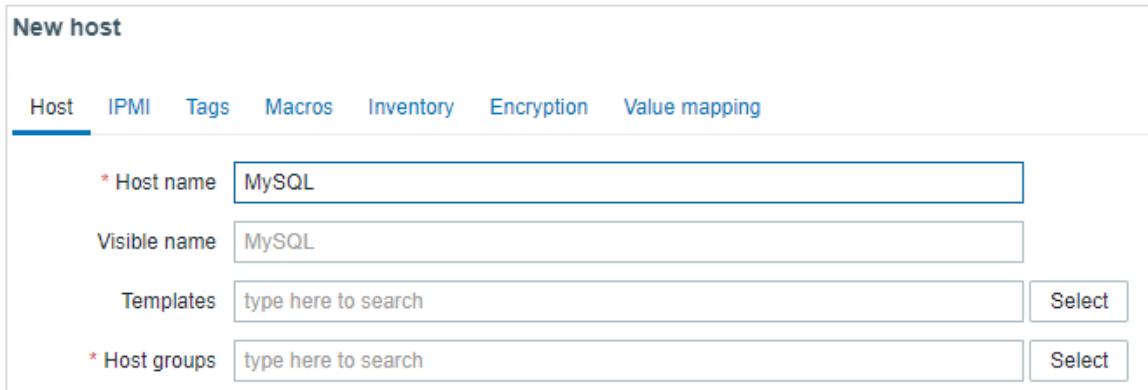
## 2.1.4 Basic Zabbix Operations

Step 1 Add a host.

In the **Monitoring** area on the Zabbix home page, click **Hosts**. The following page is displayed:



Click **Create host** in the upper right corner. On the displayed page, enter the host name as planned.



Click **Select** following **Templates** to select a template. Preset templates are available. Because the MySQL host runs in Linux and the MySQL database is running, you can enter **Operating systems** or **Databases** in the search box to filter templates. The following figure shows database-related templates. Select **MySQL by Zabbix agent2**.

### Templates

Template group

<input type="checkbox"/> Name
<input type="checkbox"/> Apache Cassandra by JMX
<input type="checkbox"/> ClickHouse by HTTP
<input type="checkbox"/> CockroachDB by HTTP
<input type="checkbox"/> GridGain by JMX
<input type="checkbox"/> Ignite by JMX
<input type="checkbox"/> MongoDB cluster by Zabbix agent 2
<input type="checkbox"/> MongoDB node by Zabbix agent 2
<input type="checkbox"/> MSSQL by ODBC
<input type="checkbox"/> MySQL by ODBC
<input type="checkbox"/> MySQL by Zabbix agent
<input checked="" type="checkbox"/> MySQL by Zabbix agent 2

You can also select multiple templates, as shown in the following figure:

### New host

Host IPMI Tags Macros Inventory Encryption Value mapping

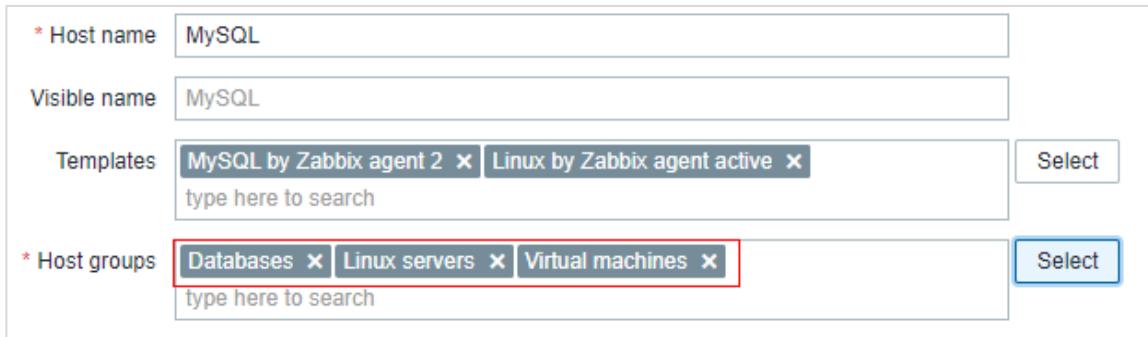
\* Host name

Visible name

Templates

\* Host groups

Specify host groups as follows:



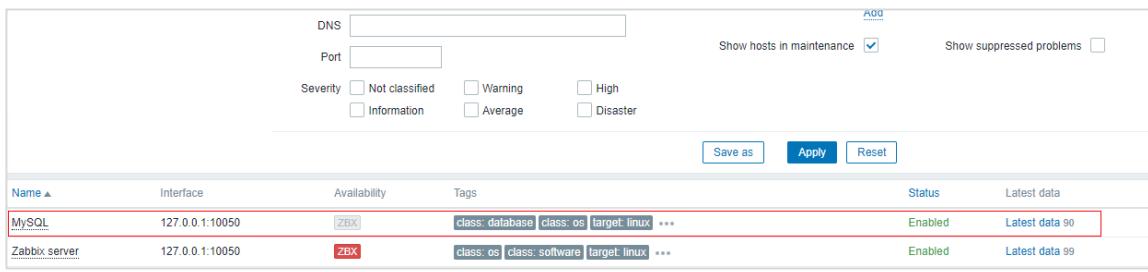
The screenshot shows the 'Hosts' configuration page in Zabbix. A new host named 'MySQL' is being created. The 'Host name' field contains 'MySQL'. The 'Visible name' field also contains 'MySQL'. Under 'Templates', 'MySQL by Zabbix agent 2' and 'Linux by Zabbix agent active' are selected. In the 'Host groups' section, 'Databases', 'Linux servers', and 'Virtual machines' are selected and highlighted with a red border. A 'Select' button is available for these groups.

Click **Add** in the **Interfaces** area, select **Agent**, and enter the agent information, as shown in the following figure:



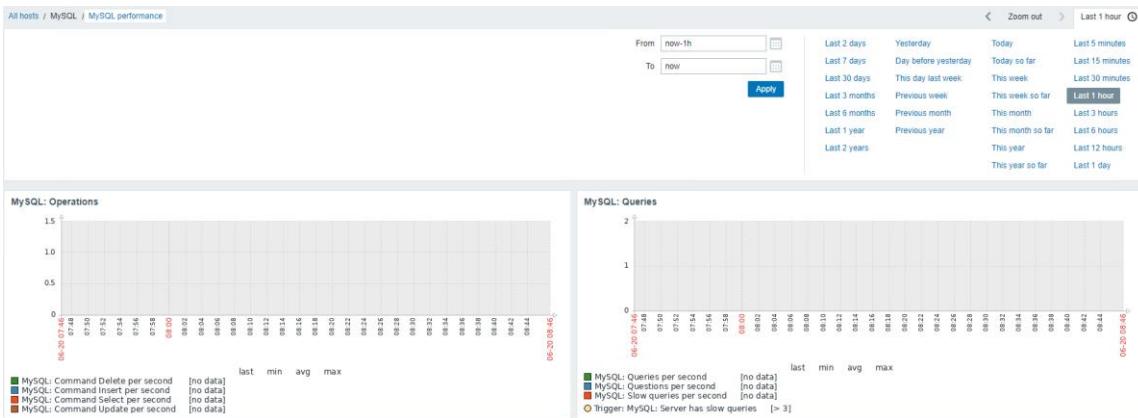
The screenshot shows the 'Interfaces' configuration for the MySQL host. An 'Agent' interface is selected with IP address '192.168.1.8'. Other fields include 'DNS name' (empty), 'Connect to' (IP), 'Port' (10050), and 'Default' (selected). A 'Rer' button is also present.

After the preceding configuration is complete, click **Add** to add the host. Then, you can view the MySQL monitoring statistics, as shown in the following figure:



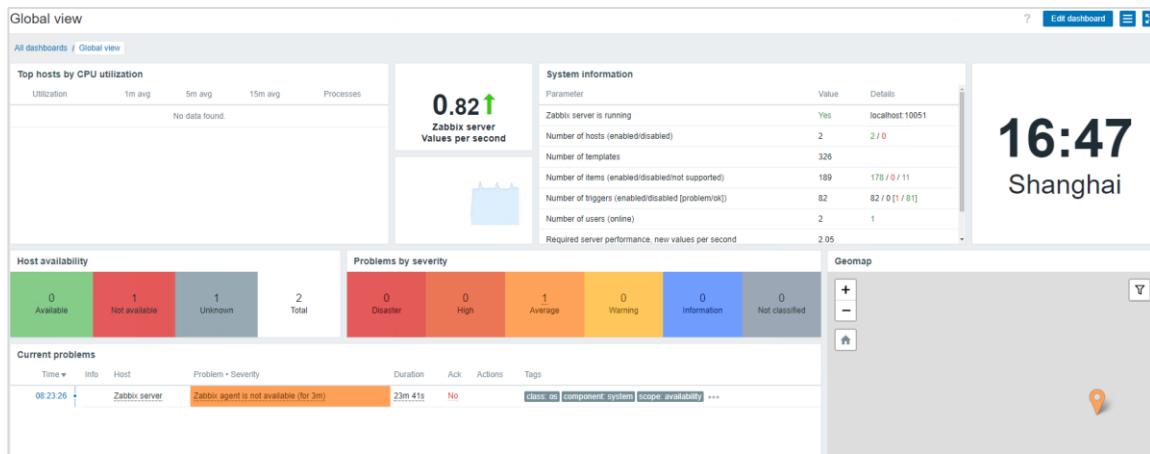
The screenshot shows the 'Hosts' list in Zabbix. It includes two hosts: 'MySQL' and 'Zabbix server'. Both hosts have an IP of '127.0.0.1:10050' and are connected via 'ZBX' (agent). They are both assigned to the 'class: database', 'class: os', and 'target: linux' tags and are marked as 'Enabled'. The 'Latest data' column shows 'Latest data 90' for MySQL and 'Latest data 99' for the Zabbix server.

You can click **Latest data**, **Graphs**, or **Dashboard** to view the monitoring statistics in respective forms. The following figure shows the dashboard view.

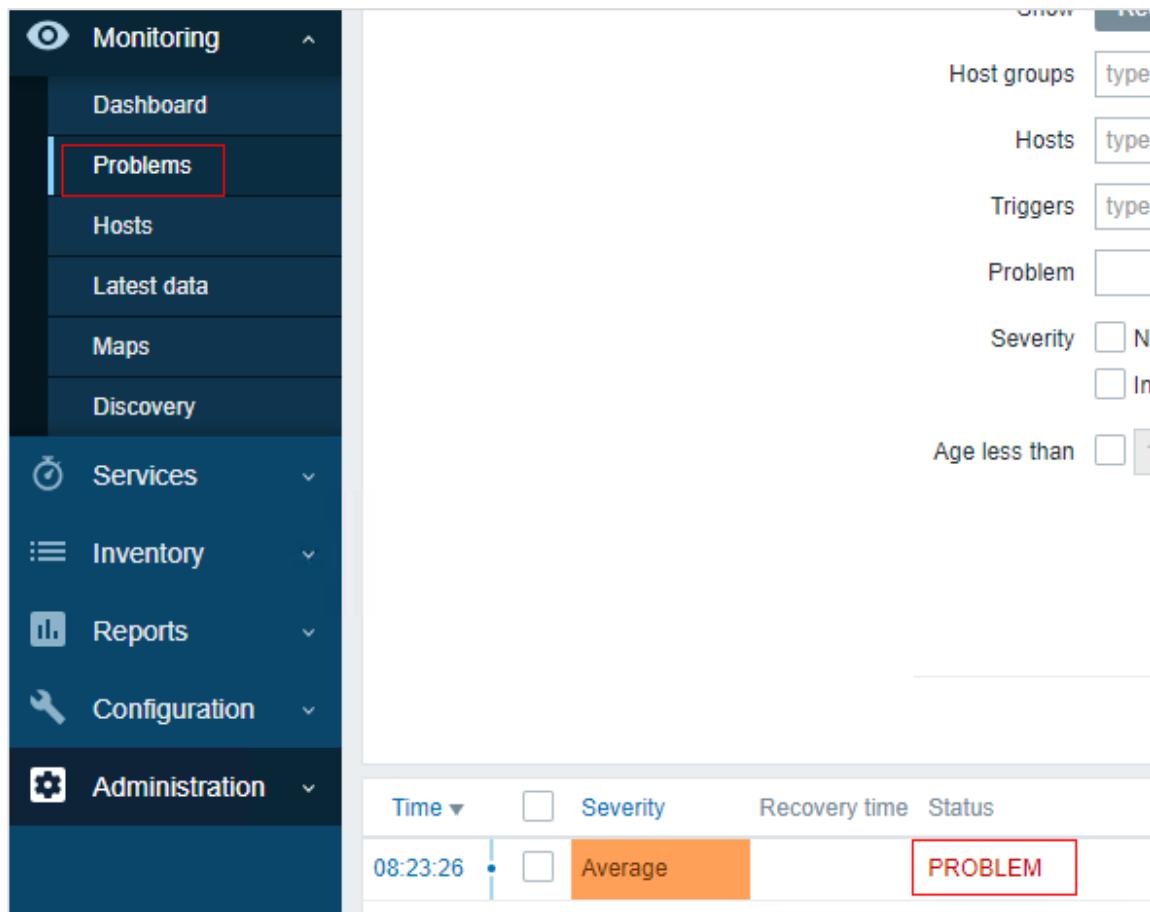


## Step 2 Check monitored hosts.

On the Zabbix home page, click **Dashboard** in the **Monitoring** area to view the global information, as shown in the following figure.



In the **Monitoring** area, click **Problems** to view information about abnormal hosts, as shown in the following figure.



Click the **Ack** status corresponding to a faulty host. The window for handling the problem is displayed, as shown in the following figures.

Time	Severity	Recovery time	Status	Info	Host	Problem	Duration	Ack	Ac
08:23:26	Average		PROBLEM		Zabbix server	Zabbix agent is not available (for 3m)	24m 46s	No	

Update problem

Problem Zabbix agent is not available (for 3m)

Message fix, close

History	Time	User	User action	Message
---------	------	------	-------------	---------

Scope  Only selected problem  
 Selected and all other problems of related triggers 1 event

Change severity  Not classified  Information  Warning  Average  High  Disaster

Suppress  Indefinitely  Until now+1d

Unsuppress

Acknowledge

Close problem

\* At least one update operation or message must exist.

Update Cancel

After the update, the status of the problem will be updated, as shown in the following figure.

Time	Severity	Recovery time	Status	Info	Host	Problem	Duration	Ack	Actions	Ta
08:23:26	<input type="checkbox"/> Average		PROBLEM		Zabbix server	Zabbix agent is not available (for 3m) 	27m 12s	No	 	C

### Step 3 Customize a monitoring item.

This following uses the monitoring of online users of the MySQL server as an example to describe how to create a custom monitoring item.

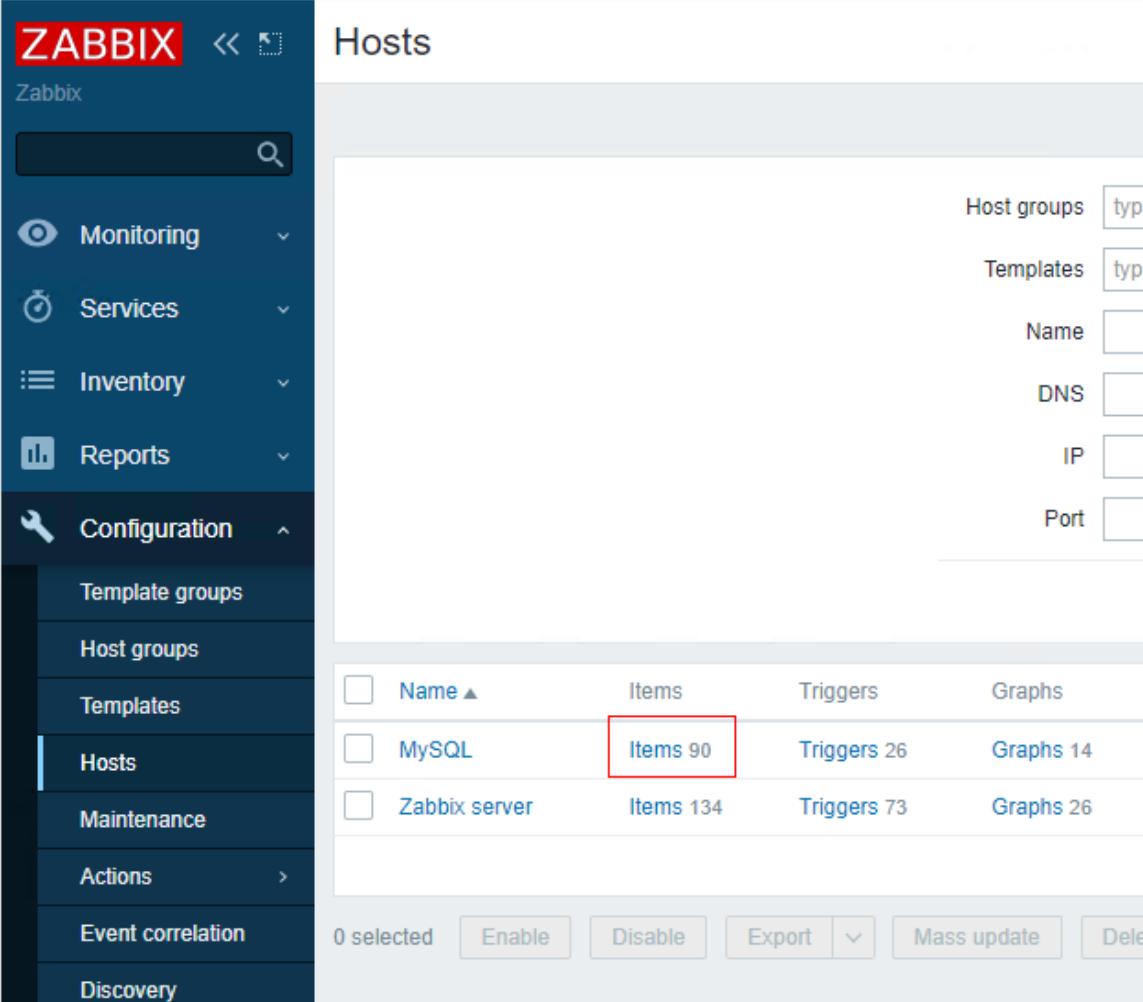
Create a **User\_MySQL.conf** configuration file in the MySQL agent installation directory, for example, **/etc/zabbix/zabbix\_agent2.d**, and enter the following content to the file:

```
UserParameter=User.MySQL,who | wc -l
```

Restart the zabbix-agent2 server and run the **zabbix\_agent2** command to check whether the value corresponding to the specified key can be obtained, as shown in the following figure.

```
[root@MySQL zabbix_agent2.d]# zabbix_agent2 -t User.MySQL  
User.MySQL [s|1]
```

If the value cannot be obtained, adjust the command based on the command output. If the value can be obtained, log in to the Zabbix server through the web interface, click **Hosts** in the **Configuration** area, and select **Items** corresponding to MySQL, as shown in the following figure.



Name	Items	Triggers	Graphs
MySQL	Items 90	Triggers 26	Graphs 14
Zabbix server	Items 134	Triggers 73	Graphs 26

On the displayed page, click **Create item** in the upper right corner and then set parameters as planned.

* Name	MySQL-online-user		
Type	Zabbix agent		
* Key	User.MySQL		
Type of information	Numeric (unsigned)		
* Host interface	10050		
Units	192.168.1.8:10050		
* Update interval	1m		
Custom intervals	Type	Interval	Period
	Flexible	Scheduling	50s
			1-7,00:00-24:00
			<a href="#">Remove</a>
<a href="#">Add</a>			
* History storage period	Do not keep history	Storage period	90d
* Trend storage period	Do not keep trends	Storage period	365d
Value mapping	type here to search		
Populates host inventory field	-None-		
Description	<div style="border: 1px solid #ccc; height: 100px; width: 100%;"></div>		
Enabled	<input checked="" type="checkbox"/> <a href="#">Add</a> <a href="#">Test</a> <a href="#">Cancel</a>		

After the configuration is complete, click **Test**. On the test page, click **Get value and test** to check whether the corresponding value can be obtained, as shown in the following figure.

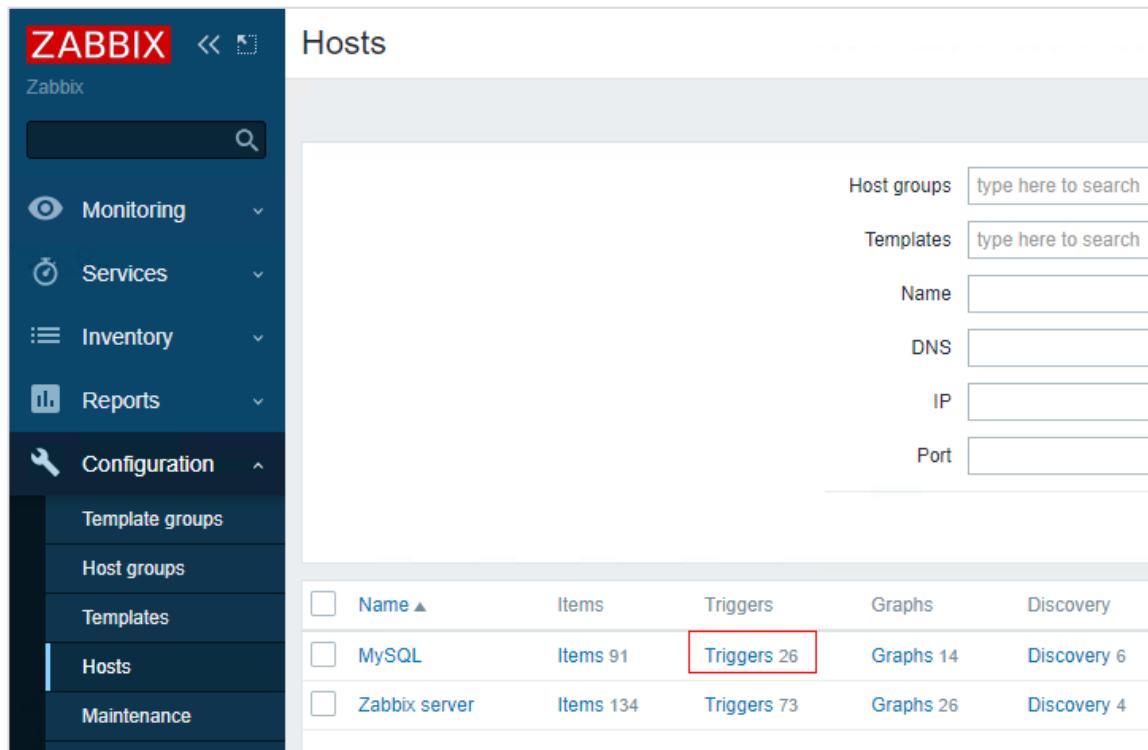
Test item	
Get value from host	<input checked="" type="checkbox"/>
* Host address	192.168.1.8
Port	10050
Proxy	(no proxy)
<a href="#">Get value</a>	
Value	3
Time now	
<input type="checkbox"/> Not supported	
Previous value	
Prev. time	
End of line sequence	<input type="radio"/> LF <input checked="" type="radio"/> CRLF
<a href="#">Get value and test</a> <a href="#">Cancel</a>	

If the returned value is normal, close the page and click **Add**.

Note: Zabbix has preset items for monitoring online users. You can refer to their configurations to perform this lab.

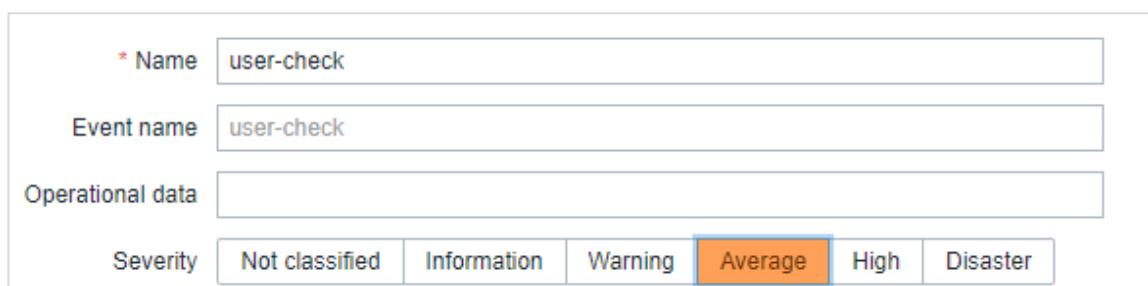
#### Step 4    Customize a trigger.

In the **Configuration** area, click **Hosts** and select **Triggers** corresponding to MySQL, as shown in the following figure.

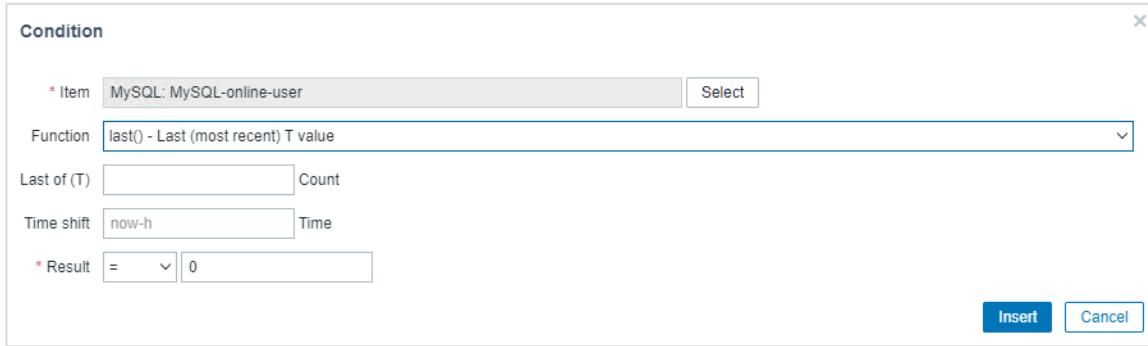


Name	Items	Triggers	Graphs	Discovery
MySQL	Items 91	Triggers 26	Graphs 14	Discovery 6
Zabbix server	Items 134	Triggers 73	Graphs 26	Discovery 4

On the displayed page, click **Create trigger** in the upper right corner. Enter the trigger name and severity as planned.

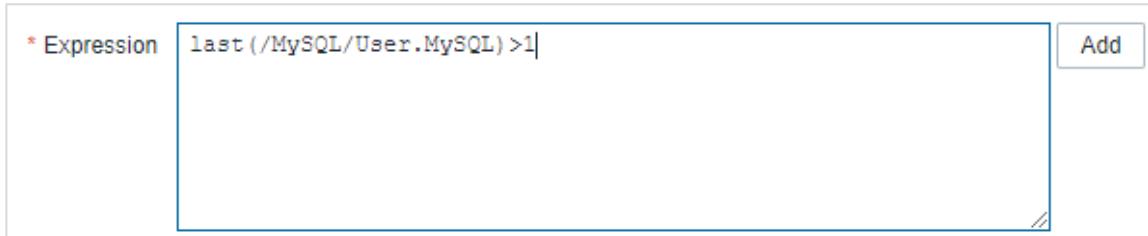


In the **Expression** area, click **Add** to add an expression as follows:



The screenshot shows the 'Condition' tab of a Zabbix trigger configuration dialog. The 'Item' field is set to 'MySQL: MySQL-online-user'. The 'Function' dropdown is set to 'last() - Last (most recent) T value'. The 'Last of (T)' dropdown is set to 'Count'. The 'Time shift' dropdown is set to 'now-h'. The 'Result' dropdown is set to '=' with a value of '0'. At the bottom right are 'Insert' and 'Cancel' buttons.

After the configuration is complete, click **Insert**. The expression is added, as shown in the following figure.



The screenshot shows the 'Expression' tab of the Zabbix trigger configuration dialog. The 'Expression' field contains the text 'last(/MySQL/User.MySQL)>1'. At the top right is an 'Add' button.

Click **Add** to create the trigger.

## 2.2 Quiz

- What are the differences between the active and passive modes of Zabbix?

Answer: The passive mode is the default mode of Zabbix. In this mode, the server polls the agent status. In active mode, the agent proactively reports information to the server, which reduces the load of the server and thus accelerates server response.

Huawei openEuler Certification Training

# HCIP-openEuler

## Lab Guide

Issue: 1.0



Huawei Technologies Co., Ltd.

**Copyright © Huawei Technologies Co., Ltd. 2024. All rights reserved.**

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

### **Trademarks and Permissions**



HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

### **Notice**

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

## **Huawei Technologies Co., Ltd.**

Address:      Huawei Industrial Base Bantian, Longgang Shenzhen 518129  
                  People's Republic of China

Website:      <http://e.huawei.com>

## Huawei Certification System

Huawei Certification is an integral part of the company's Platform + Ecosystem strategy. It supports the development of ICT infrastructure that features Cloud-Pipe-Device synergy. Our certification is always evolving to reflect the latest trends in ICT development. Huawei Certification consists of three categories: ICT Infrastructure Certification, Basic Software & Hardware Certification, and Cloud Platform & Services Certification, making it the most extensive technical certification program in the industry.

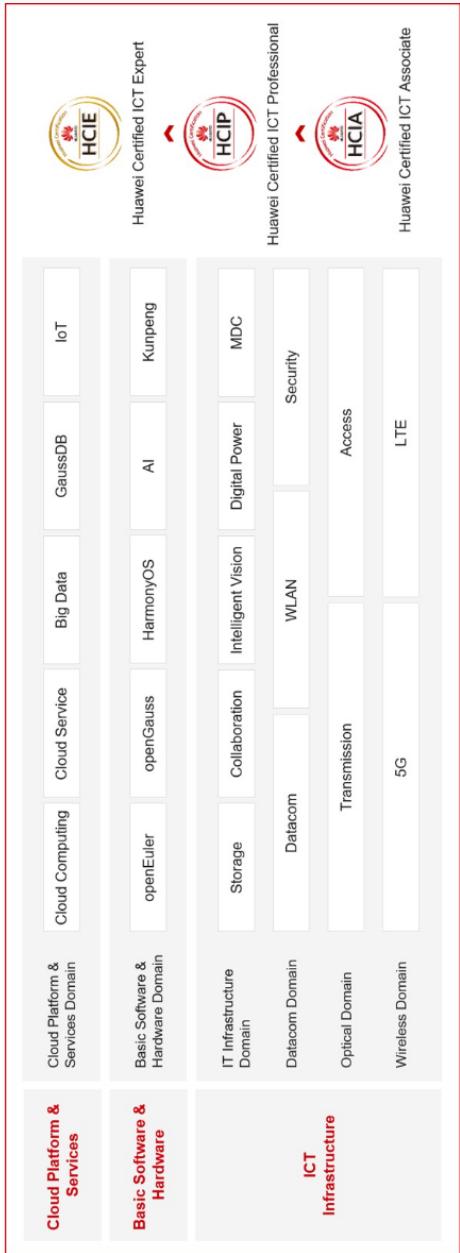
Huawei offers three levels of certification: Huawei Certified ICT Associate (HCIA), Huawei Certified ICT Professional (HCIP), and Huawei Certified ICT Expert (HCIE).

Our programs cover all ICT fields and follow the industry's trend of ICT convergence. With our leading talent development system and certification standards, we are committed to fostering new digital ICT talent and building a sound ICT talent ecosystem.

HCIP-openEuler is mainly for frontline engineers from Huawei and representative offices and readers who wish to learn openEuler O&M technologies. HCIP-openEuler certification covers common openEuler enterprise service management, openEuler HA cluster architecture, openEuler storage management, openEuler automated O&M, Linux shell scripts, openEuler system security hardening, and openEuler system monitoring.

Huawei certification helps you unlock opportunities to advance your career and take one more step towards the top of the industry.

## Huawei Career Certification



# About This Document

---

## Overview

This document is an HCIP-openEuler certification training course and is intended for trainees who are going to take the HCIP-openEuler exam or readers who want to learn how to build enterprise services, shell scripts, or perform automated O&M using Zabbix or Salt on openEuler and other Linux distributions.

## Description

This lab guide introduces a comprehensive practice, which uses Ansible to set up an enterprise website. The website can access static pages and internal blog systems. Static data of the website is stored in the shared storage provided by GlusterFS, and dynamic data is stored in the MySQL database. The domain name of the website is [www.test.com](http://www.test.com). Therefore, you need to configure the DNS to resolve the domain name to the IP address of the web server. To monitor the status of each host, you need to also install and configure Zabbix.

## Background Knowledge Required

This course is for Huawei's basic certification. To better understand this course, familiarize yourself with the following:

- Have basic Linux knowledge. You are advised to complete HCIA-openEuler learning and pass the HCIA-openEuler certification exam.

# Contents

---

<b>About This Document .....</b>	<b>3</b>
Overview .....	3
Description .....	3
Background Knowledge Required .....	3
<b>1 Comprehensive Practice – Preparation .....</b>	<b>1</b>
1.1 Environment Planning and Resource Preparation .....	1
1.1.1 Environment Planning.....	1
1.1.2 Resource Preparation .....	2
1.2 Ansible Controller Configurations and Basic Host Configurations .....	2
1.2.1 Ansible Installation and Basic Configurations.....	2
1.2.2 Basic Configurations .....	3
<b>2 Comprehensive Practice – Implementation .....</b>	<b>5</b>
2.1 Basic Software Installation .....	5
2.2 Service Configurations.....	8
2.2.1 MySQL Active/Standby Cluster Configurations.....	8
2.2.2 GlusterFS Cluster Setup .....	11
2.2.3 Apache Service Configurations .....	14
2.2.4 Nginx + Keepalived + LVS Cluster Configurations .....	17
2.2.5 DNS Configurations .....	20
2.2.6 Zabbix Configurations .....	20

# 1

## Comprehensive Practice – Preparation

### 1.1 Environment Planning and Resource Preparation

#### 1.1.1 Environment Planning

The following table lists required ECS addresses and host names.

Service	Host	IP Address	Description
LVS + Keepalived cluster Floating IP address: 192.168.1.10	LVS-01	192.168.1.11	Service plane
		10.0.0.11	Data and storage planes
	LVS-02	192.168.1.12	Service plane
		10.0.0.12	Data and storage planes
Nginx cluster	Nginx-01	10.0.0.13	Service plane
	Nginx-02	10.0.0.14	Service plane
GlusterFS cluster Floating IP address: 10.0.0.20	Cluster-01	10.0.0.21	Storage plane
	Cluster-02	10.0.0.22	Storage plane
	Cluster-03	10.0.0.23	Storage plane
MySQL cluster	MySQL-01	10.0.0.31	Data plane
	MySQL-02	10.0.0.32	Data plane
Apache cluster	Apache-01	10.0.0.41	Data and storage planes
	Apache-02	10.0.0.42	Data and storage planes
DNS	DNS	192.168.1.13	Service plane
Ansible	Ansible	192.168.1.50	Management plane
Zabbix	Zabbix	192.168.1.15	Service plane
		10.0.0.15	Management plane

The mask of all addresses is 24 bits. The gateway of 192.168.1.0/24 is 192.168.1.1. No gateway is set for the 10.0.0.0/24 network segment.

In addition, an Elastic Volume Service (EVS) drive with at least 11 GB capacity must be attached to the host in the GlusterFS cluster.

## 1.1.2 Resource Preparation

Purchase ECSs on Huawei Cloud based on the following plan:

ECS Usage	Specification	Quantity	Description
Ansible	1 vCPUs   2 GiB   s7.medium.2, single NIC	1	Ansible installation
LVS cluster	2 vCPUs   4 GiB   s7.large.2, dual NICs	2	Layer-4 proxy
Nginx cluster	1 vCPUs   2 GiB   s7.medium.2, single NIC	2	Layer-7 proxy
GlusterFS cluster	1 vCPUs   2 GiB   s7.medium.2, single NIC	3	
MySQL cluster	1 vCPUs   2 GiB   s7.medium.2, single NIC	2	Database for all applications
Apache cluster	1 vCPUs   2 GiB   s7.medium.2, single NIC	2	Web service
DNS	1 vCPUs   2 GiB   s7.small.1, single NIC	1	
Zabbix	1 vCPUs   2 GiB   s7.medium.2, dual NICs	1	

## 1.2 Ansible Controller Configurations and Basic Host Configurations

### 1.2.1 Ansible Installation and Basic Configurations

Install Ansible and perform basic configurations by following the instructions provided in section 1.1 in *Automation Management*.

The following lists the configured hosts for reference:

```
[Apache]
10.0.0.41 host=01
10.0.0.42 host=02
[Nginx]
10.0.0.13 host=01
10.0.0.14 host=02
[Gluster]
10.0.0.21 host=01
```

```
10.0.0.22 host=02
10.0.0.23 host=03
[MySQL]
10.0.0.31 host=01
10.0.0.32 host=02
[keepalive]
192.168.1.[11:12]
10.0.0.[21:23]
[lvs]
192.168.1.[11:12]
[dns]
192.168.1.13 host=dns
[zabbix]
192.168.1.15 host=zabbix
```

After the configuration is complete, use the ping module to test whether the communication between the Ansible controller and all services is normal by running the following command:

```
[root@Ansible ~]# ansible all -m ping
```

## 1.2.2 Basic Configurations

### Step 1 Change ECS host names.

Compile playbook and change the host names of all ECSs to the planned values. The following playbook content is for reference only:

```
---
- hosts: Nginx
  remote_user: root
  gather_facts: no

  tasks:
    - name: set hostname for Nginx
      hostname:
        name=Nginx-{{ host }}

- hosts: MySQL
  remote_user: root
  gather_facts: no

  tasks:
    - name: set hostname for MySQL
      hostname:
        name=MySQL-{{ host }}

- hosts: Gluster
  remote_user: root
  gather_facts: no

  tasks:
    - name: set hostname for Gluster
      hostname:
```

```
name=Gluster-{{ host }}  
  
- hosts: dns:zabbix  
  remote_user: root  
  gather_facts: no  
  
  tasks:  
    - name: set hostname for dns and zabbix  
      hostname:  
        name={{ host }}
```

# 2 Comprehensive Practice – Implementation

## 2.1 Basic Software Installation

Step 1 Install and configure Apache and PHP components.

Compile playbook to install Apache and PHP components on Apache1 and Apache2. The following information is for reference:

```
---
- hosts: Apache
  remote_user: root
  gather_facts: no

  tasks:
    - name: install httpd
      yum:
        name: httpd
        state: present
    - name: enable and start httpd
      service:
        name: httpd
        state: started
        enabled: yes
    - name: install php
      yum:
        name: php
        state: present
    - name: install php-mysqlnd
      yum:
        name: php-mysqlnd
        state: present
```

Step 2 Install a MySQL database.

Compile playbook to install MySQL-related components on MySQL1 and MySQL2. The following information is for reference:

```
---
- hosts: Mysql
  remote_user: root
  gather_facts: no
```

```
tasks:
- name: install mysql
  yum:
    name: mysql-server
    state: present
- name: enable and start mysql
  service:
    name: mysqld
    state: started
    enabled: yes
```

### Step 3 Install Keepalived.

Compile playbook to install Keepalived components on Nginx1, Nginx2, Gluster1, Gluster2, and Gluster3. The following information is for reference:

```
---
- hosts: Nginx:Gluster
  remote_user: root
  gather_facts: no

  tasks:
- name: install keepalived
  yum:
    name: keepalived
    state: present
```

- Question: In the preceding tasks, why does not start Keepalived after it is installed?

Answer: After Keepalived is installed, the configuration files of Keepalived on all hosts are the same. If Keepalived is started at this time, the startup fails. Therefore, you are advised to start Keepalived after it is configured.

### Step 4 Install Nginx.

Compile playbook to install Nginx components on Nginx1, Nginx2, and Zabbix. The following information is for reference:

```
---
- hosts: Nginx:zabbix
  remote_user: root
  gather_facts: no

  tasks:
- name: install nginx
  yum:
    name: nginx
    state: present
- name: enable and start nginx
  service:
    name: nginx
    state: started
    enabled: yes

- hosts: zabbix
```

```
remote_user: root
gather_facts: no

tasks:
- name: install php
  yum:
    name: php
    state: present
- name: config port of php
  lineinfile:
    path: /etc/php-fpm.d/www.conf
    insertafter: "listen.allowed_clients = 127.0.0.1"
    line: "listen = 9000"
- name: enable and start php
  service:
    name: php-fpm
    state: started
    enabled: yes
```

## Step 5 Install GlusterFS.

Compile playbook to install GlusterFS components on Gluster-01, Gluster-02, and Gluster-03. The following information is for reference:

```
---
- hosts: Gluster
  remote_user: root
  gather_facts: no

  tasks:
- name: install glusterfs-server
  yum:
    name: glusterfs-server
    state: present
- name: enable and star glusterfs-server
  service:
    name: glusterd
    state: started
    enabled: yes
```

## Step 6 Install the DNS service.

Compile playbook to install bind-related components on the DNS. The following information is for reference:

```
---
- hosts: dns
  remote_user: root
  gather_facts: no

  tasks:
- name: install dns
  yum:
    name: bind
```

```
state: present
- name: enable and start named
  service:
    name: named
    state: started
    enabled: yes
```

## 2.2 Service Configurations

Ansible does not have a dedicated module for configuring the database. Therefore, you need to log in to the data host to modify some configurations.

### 2.2.1 MySQL Active/Standby Cluster Configurations

Step 1 Initialize the database.

Use playbook to initialize the database and set the password of the **root** user to **Huawei@123**. The following information is for reference:

```
---
- hosts: Mysql
  remote_user: root
  gather_facts: no

  tasks:
    - name: set password for root
      command: mysql -e "alter user root@'localhost' identified by 'Huawei@123';"
```

Step 2 Modify the configuration file of the active database.

Set **10.0.0.31** as the master node of the MySQL cluster using playbook. The following information is for reference:

```
---
- hosts: 10.0.0.31
  remote_user: root
  gather_facts: no

  tasks:
    - name: create user for replication
      command: mysql -uroot -p"Huawei@123" -e "create user slave identified with
mysql_native_password by 'Huawei@123';"
    - name: grant replication for slave
      command: mysql -uroot -p"Huawei@123" -e "GRANT REPLICATION SLAVE ON *.* to
'slave'@'%';"
    - name: enable privileges
      tags: master
      command: mysql -uroot -p"Huawei@123" -e "FLUSH PRIVILEGES;"
    - name: config master
      lineinfile:
        path: /etc/my.cnf
        line: "{{ item }}"
```

```
state: present
with_items:
  - 'server-id=1'
  - 'log-bin=/var/lib/mysql/binlog'
notify: restart mysqld

handlers:
  - name: restart mysqld
    service:
      name: mysqld
      state: restarted
```

Run the following command to view the current binary log name and offset of the primary service:

```
ansible 10.0.0.31 -a 'mysql -uroot -p"Huawei@123" -e "show master status;"'
```

See the following figure.

```
[root@Ansible config]# ansible 10.0.0.31 -a 'mysql -uroot -p"Huawei@123" -e "show master status;"'
10.0.0.31 | CHANGED | rc=0 >>
File          Position        Binlog_Do_DB     Binlog_Ignore_DB   Executed_Gtid_Set
binlog.000002  157           mysql: [Warning] Using a password on the command line interface can be insecure.
[root@Ansible config]#
```

### Step 3    Modify the configuration file of the standby database.

Set **10.0.0.32** as the slave node of the MySQL cluster using playbook. The following information is for reference:

```
---
- hosts: 10.0.0.32
  remote_user: root
  gather_facts: no

  tasks:
    - name: config slave
      lineinfile:
        path: /etc/my.cnf
        line: "{{ item }}"
        state: present
      with_items:
        - 'server-id=2'
        - 'log-bin=/var/lib/mysql/binlog'
      notify: restart slave
    - name: choose master
      tags: master
      command: mysql -uroot -p"Huawei@123" -e "CHANGE MASTER TO
MASTER_HOST='10.0.0.31',MASTER_PORT=3306,MASTER_USER='slave',MASTER_PASSWORD='Huawei
@123',MASTER_LOG_FILE='binlog.000002',MASTER_LOG_POS=157;"
    - name: start slave
      tags: start
      command: mysql -uroot -p"Huawei@123" -e "start slave;"

  handlers:
    - name: restart slave
      service:
        name: mysqld
```

```
state: restarted
```

Run the following command to check the slave status:

```
ansible 10.0.0.32 -a 'mysql -uroot -p"Huawei@123" -e "show slave status\G;'''
```

See the following figure.

```
[root@Ansible config]# ansible 10.0.0.32 -a 'mysql -uroot -p"Huawei@123" -e "show slave status\G;'''  
10.0.0.32 | CHANGED | rc=0 >>  
***** 1. row *****  
Slave_IO_State: Waiting for source to send event  
    Master_Host: 10.0.0.31  
    Master_User: slave  
    Master_Port: 3306  
    Connect_Retry: 60  
    Master_Log_File: binlog.000002  
    Read_Master_Log_Pos: 2477  
    Relay_Log_File: Mysql-02-relay-bin.000002  
    Relay_Log_Pos: 490  
    Relay_Master_Log_File: binlog.000002  
    Slave_IO_Running: Yes  
    Slave_SQL_Running: Yes  
    Replicate_Do_DB:  
    Replicate_Ignore_DB:  
    Replicate_Do_Table:  
    Replicate_Ignore_Table:  
    Replicate_Wild_Do_Table:  
    Replicate_Wild_Ignore_Table:
```

If the status is **Yes**, the MySQL active/standby cluster is successfully created. Otherwise, the creation fails. For details about the failure cause, see **Last\_IO\_Error** in the command output.

#### Step 4 Create the database and user required by WordPress.

Run the following command to create the database required by WordPress and set the name of the created database to **WP** as planned:

```
ansible 10.0.0.31 -a 'mysql -uroot -p"Huawei@123" -e "create database WP character set = utf8mb4;'''
```

```
[root@Ansible config]# ansible 10.0.0.31 -a 'mysql -uroot -p"Huawei@123" -e "create database WP character set = utf8mb4;'''  
10.0.0.31 | CHANGED | rc=0 >>  
mysql: [Warning] Using a password on the command line interface can be insecure.
```

After the creation is complete, check whether the WP is synchronized on the standby node.

```
ansible 10.0.0.32 -a 'mysql -uroot -p"Huawei@123" -e "show databases"'
```

```
[root@Ansible config]# ansible 10.0.0.32 -a 'mysql -uroot -p"Huawei@123" -e "show databases"'  
10.0.0.32 | CHANGED | rc=0 >>  
Database  
WP  
information_schema  
mysql  
performance_schema  
sysmysql: [Warning] Using a password on the command line interface can be insecure.  
[root@Ansible config]#
```

If the synchronization is successful, the database is successfully configured.

Finally, log in to 10.0.0.31 and run the following commands to create the user required by WordPress:

```
mysql> CREATE USER wp@'%' identified by 'Huawei@123';  
mysql> GRANT ALL PRIVILEGES ON WP.* TO 'wp'@'%';
```

```
| mysql> FLUSH PRIVILEGES;
```

## 2.2.2 GlusterFS Cluster Setup

### Step 1 Create partitions required by the GlusterFS cluster.

Create two partitions **vdb1** and **vdb2**, both with 5 GB capacity, on the GlusterFS node. Format the partitions as xfs and mount them to **/mnt/point1** and **/mnt/point2** respectively. **point1** and **point2** are storage blocks. For details about the playbook content, see the following:

```
---
```

```
- hosts: Gluster
  remote_user: root
  gather_facts: no

  tasks:
    - name: create mount brick1
      file:
        path: /mnt/point1
        state: directory
    - name: create mount brick2
      file:
        path: /mnt/point2
        state: directory
    - name: install parted
      yum:
        name: parted
        state: present
    - name: create part1
      parted:
        device: /dev/vdb
        number: 1
        part_end: 5GiB
        state: present
    - name: create part2
      parted:
        device: /dev/vdb
        number: 2
        part_start: 5GiB
        part_end: 10GiB
        state: present
    - name: install xfsprogs
      yum:
        name: xfsprogs
        state: present
    - name: format vdb1
      filesystem:
        dev: /dev/vdb1
        fstype: xfs
        force: yes
    - name: format vdb2
      filesystem:
```

```
dev: /dev/vdb2
fstype: xfs
force: yes
- name: mount vbd1
  mount:
    src: /dev/vdb1
    path: /mnt/point1
    fstype: xfs
    state: mounted
- name: mount vbd2
  mount:
    src: /dev/vdb2
    path: /mnt/point2
    fstype: xfs
    state: mounted
- name: config hosts
  lineinfile:
    path: /etc/hosts
    line: "{{ item }}"
    state: present
  with_items:
    - '10.0.0.21 Gluster-01'
    - '10.0.0.22 Gluster-02'
    - '10.0.0.23 Gluster-03'
```

## Step 2 Create a GlusterFS cluster.

Use Keepalived to configure three GlusterFS nodes as an HA cluster. First, create the jinja2 template corresponding to the Keepalived configuration file. For details, see the following content:

```
! Configuration File for keepalived

global_defs {
    router_id {{ ansible_fqdn }}
}

vrrp_instance Nginx {
    state {{ role }}
    interface ens3
    virtual_router_id 52
    priority {{ priority }}
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        10.0.0.20/24
    }
}
```

Use playbook to upload the template to each node and start the Keepalived service. Refer to the following to configure playbook:

```
---
- hosts: 10.0.0.21
  remote_user: root
  vars:
    - role: MASTER
    - priority: 255

  tasks:
    - name: upload configuration to glusterfs
      template: src=/root/yaml/file/keepalived.conf.j2 dest=/etc/keepalived/keepalived.conf
    - name: restart keepalived
      service:
        name: keepalived
        state: started
        enabled: yes

- hosts: 10.0.0.22
  remote_user: root
  vars:
    - role: BACKUP
    - priority: 200

  tasks:
    - name: upload configuration to glusterfs
      template: src=/root/yaml/file/keepalived.conf.j2 dest=/etc/keepalived/keepalived.conf
    - name: restart keepalived
      service:
        name: keepalived
        state: started
        enabled: yes

- hosts: 10.0.0.23
  remote_user: root
  vars:
    - role: BACKUP
    - priority: 100

  tasks:
    - name: upload configuration to glusterfs
      template: src=/root/yaml/file/keepalived.conf.j2 dest=/etc/keepalived/keepalived.conf
    - name: restart keepalived
      service:
        name: keepalived
        state: started
        enabled: yes
```

### Step 3 Create logical volumes of GlusterFS.

Run the following commands to create logical volumes of GlusterFS:

```
ansible 10.0.0.21 -a "gluster peer probe Gluster-02"
ansible 10.0.0.21 -a "gluster peer probe Gluster-03"
ansible 10.0.0.21 -a "gluster volume create wp disperse 3 redundancy 1 Gluster-01:/mnt/point1
Gluster-02:/mnt/point1 Gluster-03:/mnt/point1 force"
ansible 10.0.0.21 -a "gluster volume start wp"
```

```
ansible 10.0.0.21 -a "gluster volume create image disperse 3 redundancy 1 Gluster-01:/mnt/point2  
Gluster-02:/mnt/point2 Gluster-03:/mnt/point2 force"  
ansible 10.0.0.21 -a "gluster volume start image"
```

## 2.2.3 Apache Service Configurations

### Step 1 Add PHP-related configurations.

Compile playbook and add PHP-related configurations to the Apache configuration file. For details, see the following content:

```
---  
- hosts: Apache  
  remote_user: root  
  gather_facts: no  
  
  tasks:  
    - name: config php  
      lineinfile:  
        path: /etc/httpd/conf/httpd.conf  
        insertafter: AddType application/x-gzip .gz .tgz  
        line: "    AddType application/x-httpd-php .php"
```

### Step 2 Mount logical volumes provided by GlusterFS.

According to the plan, WordPress files are stored in the **/data/wp/** directory on the Apache server, and static data is stored in the **/data/image** directory. Therefore, you need to compile playbook to create the corresponding directories and mount the logical volumes of GlusterFS to the corresponding directories as follows:

```
---  
- hosts: Apache  
  remote_user: root  
  gather_facts: no  
  
  tasks:  
    - name: create wp  
      file:  
        path: /data/wp  
        owner: apache  
        group: apache  
        recurse: yes  
        state: directory  
    - name: create image  
      file:  
        path: /data/image  
        owner: apache  
        group: apache  
        recurse: yes  
        state: directory  
    - name: install glusterfs client  
      yum:  
        name: glusterfs-client  
        state: present
```

```
- name: config hosts
  lineinfile:
    path: /etc/hosts
    line: "{{ item }}"
    state: present
  with_items:
    - '10.0.0.21 Gluster-01'
    - '10.0.0.22 Gluster-02'
    - '10.0.0.23 Gluster-03'
- name: mount glusterfs to wp
  mount:
    name: /data/wp
    src: 10.0.0.20:/wp
    fstype: glusterfs
    state: mounted
    opts: defaults,_netdev
- name: mount glusterfs to image
  mount:
    name: /data/image
    src: 10.0.0.20:/image
    fstype: glusterfs
    state: mounted
    opts: defaults,_netdev
```

### Step 3 Create a virtual host.

Create a configuration file for configuring the Apache virtual host in Ansible as follows:

```
<VirtualHost *:81>
  ServerName localhost
  DocumentRoot "/data/wp/"
  <Directory "/data/wp">
    AllowOverride None
    Require all granted
  </Directory>
</VirtualHost>
<VirtualHost *:82>
  DocumentRoot "/data/image"
  <Directory "/data/image">
    AllowOverride None
    Require all granted
  </Directory>
  ServerName localhost
</VirtualHost>
```

Use playbook to send the configuration to the Apache host and enable the corresponding port as follows:

```
---
- hosts: Apache
  remote_user: root
  gather_facts: no

  tasks:
    - name: upload configure
```

```
copy:
  src: /root/yaml/file/vhost.conf
  dest: /etc/httpd/conf.d/vhost.conf
- name: set 81 and 82
  lineinfile:
    path: /etc/httpd/conf/httpd.conf
    insertafter: Listen 80
    line: "{{ item }}"
  with_items:
    - "Listen 81"
    - "Listen 82"
- name: restart httpd
  service:
    name: httpd
    state: restarted
```

#### Step 4 Create a static page.

Log in to an Apache host, for example, host 10.0.0.41, to perform the following operations.

Create a static page for displaying images as follows:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Image display page</title>
    <style>
      img {
        max-width: 100%;
        height: auto;
      }
      .separator {
        border-top: 2px solid #000;
        margin: 20px 0;
      }
    </style>
  </head>
  <body>
    <h1>Image display page</h1>
    <ul>
      <li></li>
      <li class="separator"></li>
      <li></li>
      <li class="separator"></li>
      <li></li>
      <li class="separator"></li>
      <li></li>
    </ul>
  </body>
</html>
```

Save the file to the **/data/image/** directory and upload four images named image1.png, image2.png, image3.png, and image4.png to this directory. After the upload is complete, the files in the **/data/image/** directory are shown as follows.

```
[root@Apache-01 image]# ls  
image1.png image2.png image3.png image4.png index.html
```

- Question: Why is only one Apache host required to perform the preceding operations?

Answer: The same GlusterFS logical volumes are mounted to two Apache hosts.

Therefore, the data is consistent. Once you perform the operations on one Apache host, you can view the same data on the other Apache host.

#### Step 5 Create WordPress.

Create WordPress on one Apache host by referring to the previous content.

#### Step 6 Conduct a testing.

After all the preceding configurations are complete, use the EIP and port of the Apache host to check whether the corresponding page can be accessed. Ensure that the page is correct before performing subsequent operations.

### 2.2.4 Nginx + Keepalived + LVS Cluster Configurations

#### Step 1 Use Nginx to configure a layer-7 proxy for Apache.

On the Ansible host, create a layer-7 proxy configuration file required by Nginx and use the planned addresses to access WordPress and images as follows:

```
upstream wp {  
    server blog.test.com;  
}  
upstream image {  
    server 10.0.0.41:82;  
    server 10.0.0.42:82;  
}  
server {  
    listen 80;  
    server_name 10.0.0.12;  
    location /blog/ {  
        proxy_pass http://wp/;  
        proxy_set_header Host $host;  
        proxy_set_header X-Real-IP $remote_addr;  
        proxy_set_header X-Forwarded-For $remote_addr;  
        proxy_set_header X-Forwarded-Proto $scheme;  
        proxy_set_header X-Forwarded-Host $host;  
        proxy_set_header X-Forwarded-Port $server_port;  
    }  
    location /image/ {  
        proxy_pass http://image/;  
    }  
}
```

Note: WordPress writes the access address to the database. For example, 10.0.0.41:81 is used during WordPress installation.

Use playbook to upload the file to the Nginx host and reload the Nginx service as follows:

```
---
- hosts: Nginx
  remote_user: root

  tasks:
    - name: upload configure
      template:
        src: /root/yaml/file/proxy.conf
        dest: /etc/nginx/conf.d/proxy.conf
    - name: reload nginx
      service:
        name: nginx
        state: reloaded
```

Log in to the Nginx host and to set the gateway to the DIP of the active LVS:

```
nmcli con mod "System enp4s4" +ipv4.gateway 10.0.0.11
```

Run the following command for the configuration to take effect:

```
nmcli con down "System enp4s4" && nmcli con up "System enp4s4"
```

## Step 2 Use LVS and Keepalived to configure a layer-4 Proxy for Nginx.

Compile the Keepalived configuration file as follows:

```
! Configuration File for keepalived

global_defs {
    router_id {{ ansible_fqdn }}
}

vrrp_instance LVS {
    state {{ role }}
    interface ens3
    virtual_router_id 53
    priority {{ priority }}
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.1.10/24
    }
}
virtual_server 192.168.1.10 80 {
    delay_loop 6
    lb_algo rr
    lb_kind NAT
    persistence_timeout 50
}
```

```
protocol TCP

real_server 10.0.0.13 80 {
    weight 1
    TCP_CHECK {
        connect_timeout 3
        retry 3
        delay_before_retry 3
    }
}
real_server 10.0.0.14 80 {
    weight 2
    TCP_CHECK {
        connect_timeout 3
        retry 3
        delay_before_retry 3
    }
}
```

Compile playbook, upload the Keepalived configuration file to the corresponding host, and restart the Keepalived service as follows:

```
---
- hosts: 192.168.1.11
  remote_user: root
  vars:
    - role: MASTER
    - priority: 255

  tasks:
    - name: upload configuration to Nginx
      template: src=/root/yaml/file/lvs-keep.conf.j2 dest=/etc/keepalived/keepalived.conf
    - name: restart keepalived
      service:
        name: keepalived
        state: restarted
        enabled: yes

- hosts: 192.168.1.12
  remote_user: root
  vars:
    - role: BACKUP
    - priority: 200

  tasks:
    - name: upload configuration to Nginx
      template: src=/root/yaml/file/lvs-keep.conf.j2 dest=/etc/keepalived/keepalived.conf
    - name: restart keepalived
      service:
        name: keepalived
        state: restarted
        enabled: yes
```

## 2.2.5 DNS Configurations

### Step 1 Configure the DNS service.

In this practice, there is only one DNS server. Therefore, you can directly log in to the DNS server for configurations. According to the planning, you need to resolve [www.test.com](http://www.test.com) to **192.168.1.10**, add a record A (**blog.test.com**), and resolve it to **10.0.0.41** and **10.0.0.42**. For details about the DNS configuration file, see the following:

```
$TTL 1D
@ IN SOA master.test.com. admin.test.com. (
    0      ; serial
    1D     ; refresh
    1H     ; retry
    1W     ; expire
    3H )   ; minimum
NS      master
master A 192.168.1.10
www    CNAME main
main A 192.168.1.10
blog A 10.0.0.41
blog A 10.0.0.42
```

Supplement other configurations by referring to chapter 1 so that hosts in the **192.168.1.x/24** network segment can resolve [www.test.com](http://www.test.com) and Nginx hosts can resolve [blog.test.com](http://blog.test.com).

When WordPress is installed, it writes the address of the host where it is located to the database. You need to change the address in the database to **blog.test.com** on the **MySQL** host.

```
use WP;
update wp_options set option_value="http://blog.test.com:81" where option_name="home";
update wp_options set option_value="http://blog.test.com:81" where option_name="siteurl";
```

## 2.2.6 Zabbix Configurations

### Step 1 Install zabbix-server.

In this practice, zabbix-server is deployed in a single-node system. You need to install zabbix-server on the zabbix host based on the learned knowledge.

### Step 2 Install zabbix-agent.

Compile ansible-playbook to install zabbix-agent2 on the involved hosts as follows:

```
---
- hosts: all
  remote_user: root
  gather_facts: no

  tasks:
    - name: install zabbix-release
```

```
command: rpm -Uvh https://repo.zabbix.com/zabbix/6.2/rhel/8/x86_64/zabbix-release-6.2-3.el8.noarch.rpm
- name: install zabbix-agent2
  yum:
    name: zabbix-agent2
    state: present
- name: assign IP of zabbix_server
  replace:
    path: /etc/zabbix/zabbix_agent2.conf
    regexp: Server=127.0.0.1
    replace: Server=10.0.0.31
- name: enable zabbix-agent2
  service:
    name: zabbix-agent2
    state: started
    enabled: yes
```

### Step 3 Add hosts.

Add all hosts on the Zabbix web page. The following figure shows some host information.

glusterfs1	10.0.0.21:10050	ZBX	
mysql1	10.0.0.31:10050	ZBX	class: database   target: mysql
mysql2	10.0.0.32:10050	ZBX	class: database   target: mysql
nginx1	10.0.0.11:10050	ZBX	class: software   target: nginx
nginx2	10.0.0.12:10050	ZBX	class: software   target: nginx
Zabbix server	127.0.0.1:10050	ZBX	class: os   class: software   target: linux ...

-----End-----