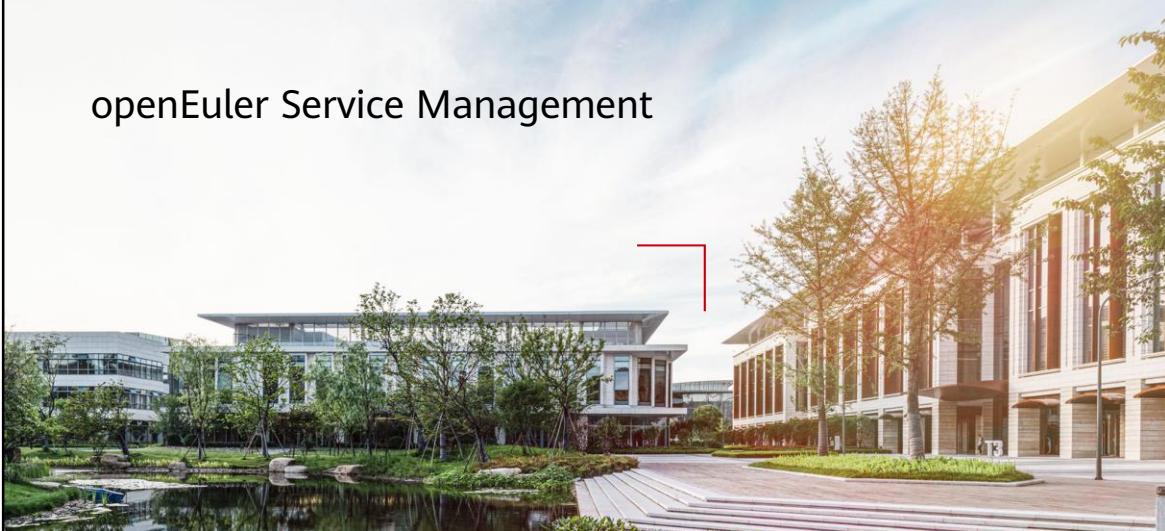


openEuler Service Management



Foreword

- Web is one of the most successful services since the birth of the Internet. It greatly promotes the development of the Internet. This course describes the mainstream open source web service applications and other related applications.

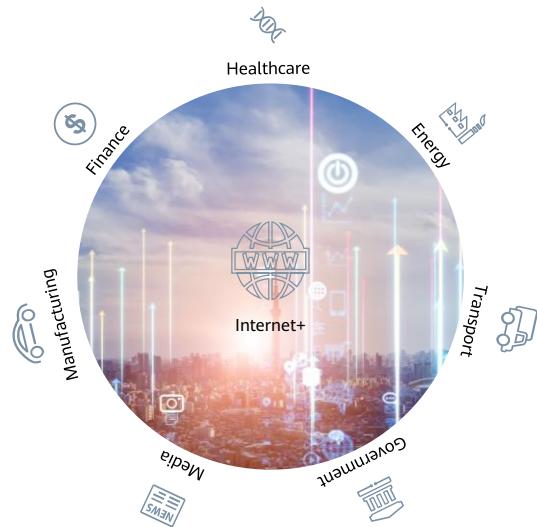
Objectives

- On completion of this course, you will understand:
 - Apache working principles, setup, and configuration.
 - NGINX working principles, setup, and configuration.
 - DNS working principles, setup, and configuration.
 - Basic operations on MySQL databases.

Contents

- 1. Web Services**
- 2. Apache Overview and Configurations
- 3. NGINX Overview and Configurations
- 4. DNS Overview and Configurations
- 5. MySQL Overview and Configurations
- 6. LAMP/LNMP

Status Quo of the Internet



5 Huawei Confidential



- <http://3ms.huawei.com/km/static/image/detail.html?fid=63932>

Internet, World Wide Web, and Web

- The World Wide Web (WWW) or simply the Web is an information system comprising all resources that can be identified by their URLs.



- Web addresses you access on the Internet start with **www**.
- Image source: <http://3ms.huawei.com/km/static/image/detail.html?fid=63987>

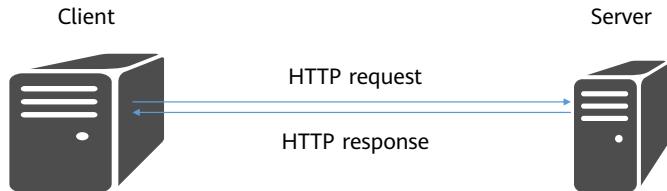
Three Elements of the World Wide Web

- Uniform Resource Identifier (URI)
 - Uniform resource locator (URL): the address of an Internet resource.
 - Uniform resource name (URN): the name of an Internet resource.
- Hypertext Markup Language (HTML)
 - The standard markup language for documents designed to be displayed in a web browser.
- Hypertext Transfer Protocol (HTTP)
 - The foundation of data communication for the World Wide Web.

- A URL enables you to locate a resource on the Internet, for example, www.huawei.com. A URN is only a resource name and its location cannot be determined. For example, a URN is used to download a P2P resource, and a magnetic link is a typical URN.
- A URL is composed of several parts:
$$<\text{scheme}>://<\text{user}>:<\text{password}>@\langle\text{host}\rangle:\langle\text{port}\rangle/\langle\text{path}\rangle;\langle\text{params}\rangle?\langle\text{query}\rangle#\langle\text{frag}\rangle$$
 - *scheme*: protocol used for accessing Internet resources, such as HTTP, HTTPS, and FTP.
 - *user*: user name for accessing Internet resources.
 - *password*: password for accessing Internet resources.
 - *host*: name or IP address of a host that provides resources on the Internet.
 - *port*: port number used by a host to provide resources.
 - *path*: path for storing the accessed resource on a host.
 - *params*: parameters carried for resource access.
 - *query*: parameters passed to a host for resource query.
 - *frag*: name of a piece or part of a resource.

HTTP Working Principle

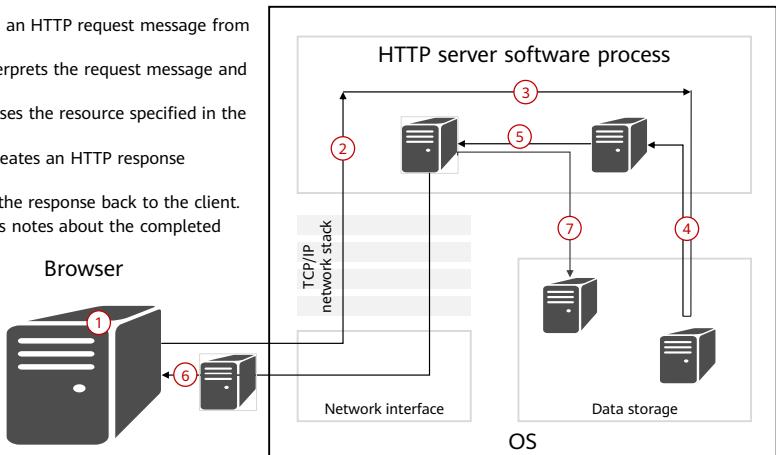
- An HTTP transaction consists of a request and a response.



- Image source: <http://3ms.huawei.com/km/static/image/detail.html?fid=62232>

Steps of an HTTP Request

1. Set up a connection: A server accepts a client connection, or closes if the client is unwanted.
2. Receive a request: The server reads an HTTP request message from the network.
3. Process the request: The server interprets the request message and takes action.
4. Access a resource: The server accesses the resource specified in the message.
5. Construct a response: The server creates an HTTP response message with the right headers.
6. Send a response: The server sends the response back to the client.
7. Log a transaction: The server places notes about the completed transaction in a log file.



9 Huawei Confidential



- The figure and description are from the *HTTP The Definitive Guide*.

Characteristics of HTTP

- Unidirectional connection: A connection request can be initiated only from a client to a server.
- Stateless protocol: Each request is independent and the connection closes once the response is complete.
 - HTTP connections are divided into two categories:
 - Persistent connections
 - Non-persistent connections
- Plaintext transmission

- Non-persistent connection: It is the default connection mode in HTTP/1.0. A connection is set up between the browser and the server to perform an HTTP job. Once the job is complete, the connection is interrupted. Non-persistent connections are recommended for websites.
- Persistent connection: It is the default connection mode since HTTP/1.1, where **Connection:keep-alive** is added to the response header. Once opened, the TCP connection for transmitting HTTP data between the browser or client and the server remain open for multiple requests/responses, avoiding resource waste when a new connection is opened for every single request/response pair. A timeout interval is set that, when expires, the connection automatically interrupts.
- Persistent connections are mainly used in frequent interactions between clients and servers, for example, database connections.

Evolution of HTTP

- HTTP/0.9
 - The 1991 prototype version of HTTP that supports only the GET method.
- HTTP/1.0
 - Launched in 1996 with the POST and HEAD methods introduced and supports only non-persistent connections.
- HTTP/1.1
 - Launched in 1997 with the PUT, PATCH, OPTIONS, and DELETE methods introduced and supports persistent connections.
- HTTP/2.0
 - Launched in 2015, derived from SPDY, and introduced functions such as header compression and server push.
 - Multiplexing multiple requests over a single TCP connection, meaning that a single connection between the client and the server can be used to serve multiple requests (fixing the head-of-line blocking problem).
- HTTP/3.0
 - Launched on June 6, 2022, and used UDP to replace TCP for lower latency.

- SPDY was developed by Google and launched in 2009. It differs from HTTP/2.0 in the following aspects:
 - HTTP/2.0 supports plaintext HTTP transmission, while SPDY forces the use of HTTPS.
 - The message header compression algorithm of HTTP/2.0 uses HPACK instead of DEFLATE, which is used by SPDY.

HTTP Message Structure

- An HTTP message consists of three parts:

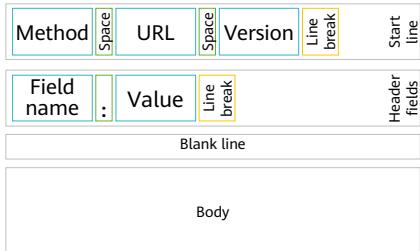
- Start line
- Header fields
- Body

| |
|-------------------------|
| GET /hello.txt HTTP/1.1 |
| Accept: text/* |
| Host: www.test.com |

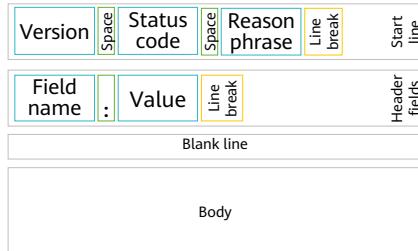
| |
|--------------------------|
| HTTP/1.1 200 OK |
| Content-type: text/plain |
| Content-length: 19 |

Hello, openEuler

Request message



Response message



- The first line of a message is the start line, indicating what to do for a request or what happened for a response.
- Zero or more header fields follow the start line. Each header field consists of a name and a value, separated by a colon (:) for easy parsing.
- After the blank line is an optional message body containing any kind of data. The request body carries data to the web server; the response body carries data back to the client. Unlike the start line and header, which are textual and structured, the body can contain arbitrary binary data, for example, images, videos, audio tracks, and software applications.

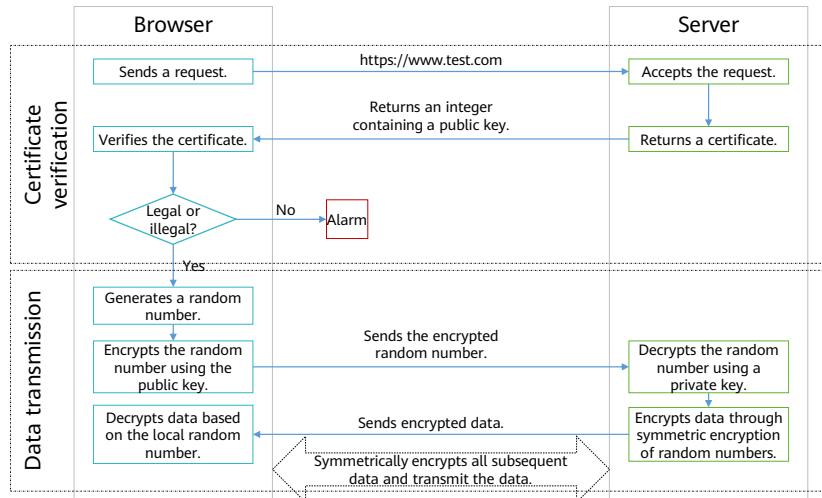
HTTP Request Methods

- GET: requests resources from the server.
- POST: sends data to the server.
- PUT: writes a document to the server.
- DELETE: deletes the resource specified in the URL.
- HEAD: returns only the headers associated with a resource, but not the body.
- TRACE: If a request initiated by a client passes through an application such as a firewall or agent, the original HTTP request is modified. TRACE enables the client to view the exact request sent to the server.
- OPTIONS: requests permitted communication options.

HTTPS

- Prevents security risks caused by HTTP plaintext transmission.
- Requires certificates from the CA.
- Works on TCP-based SSL/TLS to encrypt all contents.
- Runs on port 443 by default, which is different from HTTP.
- Has encryption/decryption overhead that slows access speed, but this impact is negligible with optimization.
- Currently, almost all websites use HTTPS.

HTTPS Session Overview

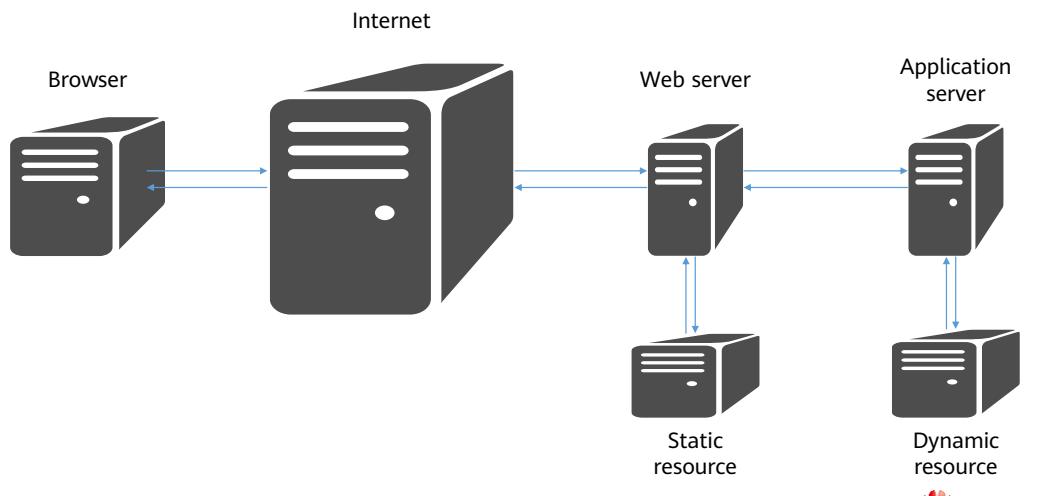


15 Huawei Confidential



- HTTPS session process:
 - The client sends available encryption methods and requests a certificate from the server.
 - The server returns the certificate and the selected encryption method to the client.
 - The client obtains and verifies the certificate in terms of:
 - Legitimacy of the source
 - Validity of content
 - Valid of the certificate
 - Whether the name of the owner is the same as that of the target host
 - The client generates a temporary session key (symmetric key), encrypts the data using the public key of the server, and sends the encrypted data to the server to complete key exchange.
 - The server uses this key to encrypt the requested resource and sends a response to the client.

One of the Most Successful HTTP Applications – Web



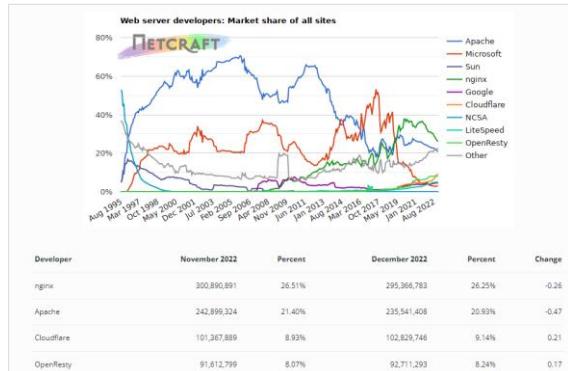
16 Huawei Confidential

 HUAWEI

- After a web address (URL) is entered in the browser, the browser uses HTTP to request the specified resource from the web server. After receiving the request, the web server returns the resource to the browser and displays the resource to the user in HTML format.

Web Servers

- A web server handles HTTP and TCP connections, manages the resources served by the web server, and provides administrative features to configure, control, and enhance the web server.
- Mainstream web servers include:
 - Nginx
 - Apache
 - IIS



17 Huawei Confidential

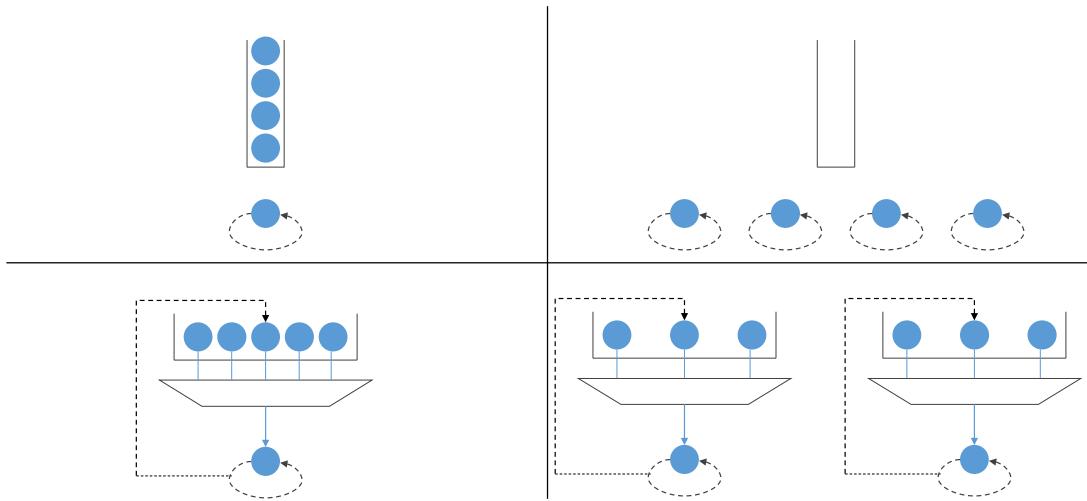


- NGINX and Apache run on the Linux OS and are the focus of subsequent learning of this course.
- IIS run on Windows.
- Source: <https://news.netcraft.com/archives/category/web-server-survey/>

Web Resources

- Web resources are classified into static and dynamic resources.
 - Static resources
 - Static resources can be directly accessed from the server in the same form as they are stored in. These include files stored on web servers, like text, HTML, image, and video files.
 - Dynamic resources
 - Resources that need to be processed by the application server in a format that can be read by users. These include search or transaction results.

How Web Servers Accept HTTP Connections

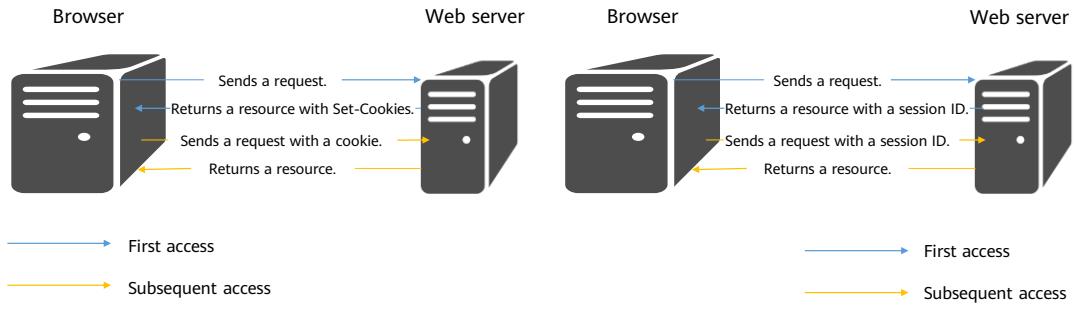


19 Huawei Confidential

- Single-threaded I/O architecture: Single-threaded web servers process one request at a time until completion. Multiple requests are responded in serial mode.
- Multithreaded I/O architecture: Multithreaded web servers start multiple processes to process requests simultaneously, with each process responds to a connection request.
- Multiplexed I/O architecture: Multiplexed web servers start a process to respond to multiple connection requests simultaneously.
- Multiplexed, multithreaded I/O architecture: Such web servers start M processes with each process responds to N connection requests to receive $M \times N$ requests simultaneously.

Cookies and Sessions

- Cookies and sessions are born to solve the problems caused by HTTP statelessness.



- HTTP is a stateless protocol. If subsequent access needs to use the resource obtained from the previous access, the client needs to retransmit the resource, causing resource waste. To address this, HTTP adds the cookie and session mechanisms to manage sessions and save the status.
- The Set-Cookie response header is used to send a cookie from the web server. After receiving the response, the browser saves the field locally. In addition, the field is constructed by the app developer on the server. When the browser accesses the web server later, the cookie is carried. If the cookie stores a large amount of data, bandwidth is wasted. In addition, if the authentication information is stored, security risks rise. Sessions are born to address such issues.
- The server generates a session ID with a value after receiving a request from the browser. The value is stored on the web server, and the session ID is returned to the browser. During subsequent access, the browser carries the session ID to the server, which then finds the corresponding value for further processing, for example, permission authentication.
- Cookies and sessions are usually used together, though not mandatory.

Web Server Response Codes

- 100-199: informational status codes
- 200-299: success status code
- 300-399: redirection status code
- 400-499: client error status code
- 500-599: server error status code
- Common status codes:
 - 200 OK indicates that the connection is normal.
 - 403 Forbidden indicates that the access is denied. In most cases, the cause is that the client does not have the permission to access the requested file or the file or directory corresponding to the accessed resource is damaged.
 - 404 Not Found indicates that the server cannot find the requested resource.
 - 503 Server Unavailable indicates that the server is not ready to handle the request from the client but can try to refresh and resend a new request.
 - 504 Gateway Timeout indicates that the server did not get a response in time from the upstream server.

- 301 Moved Permanently indicates that the requested resource has been definitively moved to the URL given by the Location headers.
- 302 Found (previously Moved Temporarily) indicates that the resource requested has been temporarily moved to the URL given by the Location headers.
- 304 Not Modified indicates that there is no need to retransmit the requested resources. This response code is sent when the client sends a conditional request but the resources on the server do not change.
- 307 Temporary Redirect indicates that the resource requested has been temporarily moved to the URL given by the Location headers.
- 401 Unauthorized indicates that the client request has not been completed because it lacks valid authentication credentials for the requested resource.
- 500 Internal Server Error indicates that an internal server error occurred.
- 502 Bad Gateway indicates that the server, while acting as a gateway or proxy, received an invalid response from the upstream server.

Contents

1. Web Services
- 2. Apache Overview and Configurations**
3. NGINX Overview and Configurations
4. DNS Overview and Configurations
5. MySQL Overview and Configurations
6. LAMP/LNMP

Apache HTTP Server

- The Apache HTTP Server (or httpd) is the first project of the Apache Software Foundation that placed a solid footing for future developments.

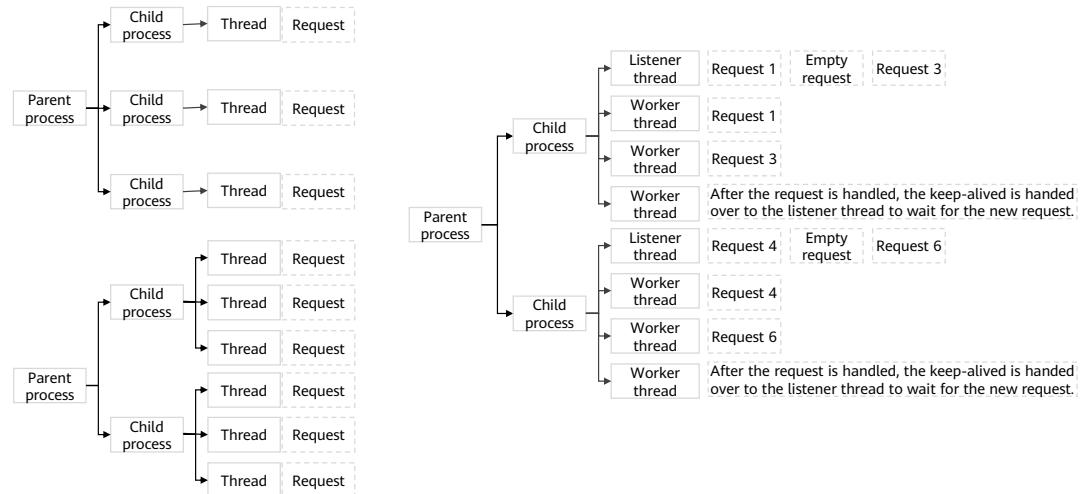
| APACHE PROJECT LIST | | | | | | | | | |
|--------------------------|-----------|------------------|------------|------------------|------------------|-----------------|--|--|--|
| Overview All Projects | | BY NAME | | | | | | | |
| BY CATEGORY | | Cocoon | Guacamole | Linkis | Petri | Submarine | | | |
| Attic | A | Commons | Gump | Logging Services | Phoenix | Subversion | | | |
| Big Data | API SIX | Community | HAWQ | Lucene | Pig | Superset | | | |
| Build Management | Accumulo | Development | HBase | MADlib | Pivot | Synapse | | | |
| Cloud | ActiveMQ | Cordova | Hadoop | MINA | Portable Runtime | SystemDS | | | |
| Content | CouchDB | Créador | Helix | MXNet | (APR) | T | | | |
| Databases | Airavata | Curator | Hive | Mahout | Portals | TVM | | | |
| FTP | Airflow | cTAKES | Hop | ManifoldCF | Pulsar | Tapestry | | | |
| Graphics | Allura | HttpComponents | Hudi | Maven | Qpid | Tcl | | | |
| HTTP | Ant | DB | Iceberg | Mesos | Ranger | Tez | | | |
| HTTP-module | Any23 | Daffodil | DataFu | Mnemonic | Rats | Thrift | | | |
| Incubating | Archiva | DataSketches | Ignite | MyFaces | RocketMQ | Tika | | | |
| JavaEE | Aries | DeltaSpike | Impala | Mynewt | Roller | TinkerPop | | | |
| Libraries | Arrow | Directory | InLong | N | Ryza | TomEE | | | |
| Mail | AsterixDB | DolphinScheduler | Incubator | NetBeans | S | Tomcat | | | |
| Mobile | Atlas | Doris | IoTDB | NIFI | SINGA | Traffic Control | | | |
| Network-client | Attic | Drill | Isis | Nutch | SIS | Turbine | | | |
| Network-server | Avro | Druid | J | Nuttx | Samza | UIMA | | | |
| OSGi | BVal | Dubbo | JMeter | O | Santuario | Uomi | | | |
| RegExp | Bahir | ECharts | JSPWiki | OFBiz | V | | | | |
| Retired | | | Jackrabbit | OODT | | | | | |

Key Features of the Apache HTTP Server

- Modular design
 - The Apache HTTP Server comprises core modules and other modules.
- Dynamic Shared Object (DSO)
 - Load/Unload DSO modules.
- Multi-Processing Module (MPM)
 - Configure the working mode of responding to multiple requests.

- Core modules refer to the modules that must be statically compiled during Apache installation and compilation, such as mod_so and mod_core. mod_core is used to process configuration directives of configuration files, and mod_so is used to dynamically load other modules.
- MPM is also considered as a core module.
- Other modules, such as mod_ssl and mod_perl, are optional. If any applications depend on them, install them. If they are not installed, Apache running is not affected.

MPMs of the Apache HTTP Server



- 1. **Prefork mode** (top left figure): Apache immediately creates multiple child processes to respond to connection requests after startup. This reduces overheads generated when child processes are frequently created and destroyed, and ensures a stable web server runtime. But child processes occupy memory resources, meaning this mode is not recommended for high-concurrency scenarios.
- 2. **Worker mode** (bottom left figure): Apache creates multiple child processes after startup, and each child process creates threads to respond to connection requests. This mode occupies fewer resources and can handle a large number of concurrent requests, however, if a child process or thread breaks down, all jobs in the child process are affected. In addition, threads are always occupied by persistent connection requests and will be released only after timeout.
- 3. **Event mode**: The latest release in Apache 2.4 and later versions. Each child process has a listener thread, which is used to solve the problem that persistent connections always occupy threads in worker mode. The listener thread accepts requests and forwards them to worker threads, and the worker threads then handle the requests. The listener thread forwards only real requests to the worker threads, but not empty threads.

Installing the Apache HTTP Server

- Installed using Yum

```
[root@apache-http mnt]# dnf install -y httpd
local
Last metadata expiration check: 0:00:02 ago on 2023-07-03 09:49:38 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
```

- Installed from the source code

- The OS has requirements on dependencies.
 - Cross-platform program development is involved.

- Cross-platform program development means that some developers in a large development team use Windows for development but others use Linux. Therefore, a unified API is required for httpd development. APR can be installed during compilation and installation. In this way, problems caused by invoking different APIs by different OSs can be avoided.

Directory Structure and Description of the Apache HTTP Server

- The configuration files of the Apache HTTP Server are located in **/etc/httpd**. The directory structure is as follows:

```
conf                                ##### Directory of the main configuration file
|   httpd.conf                         ##### Main configuration file
|   magic
conf.d                               ##### Directory of extension configuration files
|   README
|   source.conf
|   userdir.conf
|   vhost.conf
|   welcome.conf
conf.modules.d                        ##### Directory for storing the configuration files of DSO modules
|   00-base.conf                         ##### Configuration files of DSO modules
|   00-dav.conf
|   00-lua.conf
|   00-mpm.conf
|   00-optional.conf
|   00-proxy.conf
|   00-systemd.conf
|   01-cgi.conf
|   10-h2.conf
|   10-proxy_h2.conf
|   README
logs -> ../../var/log/httpd          ##### Soft link that points to the directory for storing log files
modules -> ../../usr/lib64/httpd/modules ##### Soft link that points to the directory for storing DSO modules
run -> /run/httpd                   ##### Soft link that points to the directory for storing the ID of the httpd parent process
state -> ../../var/lib/httpd         ##### Soft link file that points to the directory for storing the httpd status information
```

- Generally, the main configuration file is used to store global configurations. To modify specific parameters or functions, create an extension configuration file in the **conf.d** directory instead of modifying the main configuration file. This improves the readability of the configuration files and facilitates subsequent maintenance.

Key Parameters in the Main Configuration File of the Apache HTTP Server – Global Configurations

```
ServerRoot "/etc/httpd"  
Include conf.modules.d/*.conf  
User apache  
Group apache  
ServerAdmin root@localhost  
ServerName www.test.com:80  
<Directory />  
    AllowOverride none  
    Require all denied  
</Directory>  
DocumentRoot "/var/www/html"  
<Directory "/var/www">  
    AllowOverride None  
    Require all granted  
</Directory>  
<Directory "/var/www/html">  
    Options Indexes FollowSymLinks  
    AllowOverride None  
    Require all granted  
</Directory>  
.....(See the next page.).....
```

Root path of the configuration file
Configuration files of DSO modules
Address contained in the error message returned by the server to the client
Server name
Path for storing the home page

- **grep -Ev "^[]*#|^\\$|^#" httpd.conf** can filter out comment lines and blank lines in the main configuration file.
- The **ServerRoot** directive sets the directory where the server is located. Typically, it will contain subdirectories **conf/** and **log/**. Relative paths in other configuration directives (for example, **Include** or **LoadModule**) are considered relative to this directory.
- **Include** and **includeOptional** are module configuration files of the httpd service. There are differences between them. For example, if no corresponding configuration file is found in the corresponding directory, the system reports an error when **Include** is used but does not report an error when **IncludeOptional** is used.

Key Parameters in the Main Configuration File of the Apache HTTP Server – Global Configurations

```
.....(continued).....  
<IfModule dir_module>  
    DirectoryIndex index.html  
</IfModule>  
.....  
ErrorLog "logs/error_log"  
LogLevel warn  
<IfModule log_config_module>  
    LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" combined  
    LogFormat "%h %l %u %t \"%r\" %>s %b" common  
<IfModule logio_module>  
    LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %l %O" combinedio  
</IfModule>  
    CustomLog "logs/access_log" combined  
</IfModule>  
<IfModule alias_module>  
    ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"  
</IfModule>  
.....  
IncludeOptional conf.d/*.conf  
#### Specified home page  
#### Path for storing error logs  
#### Configuration file of a DSO module
```

- **grep -Ev "^[]*#|^\\$|^#" httpd.conf** can filter out comment lines and blank lines in the main configuration file.
- **Include** and **includeOptional** are module configuration files of the httpd service. There are differences between them. If there is no configuration file found in the directory, the system reports an error when **Include** is used but does not report an error when **IncludeOptional** is used.

Basic Configurations of the Apache HTTP Server – Home Page

- Specify the listening IP address and port of the server.
 - **Listen 80**
- Specify the directory for storing the home page.
 - **DocumentRoot "/var/www/html"**
- Specify the default page.
 - **DirectoryIndex index.html**

- If no IP address is specified when the port is specified, the httpd service can be accessed through any IP address of the server. If an IP address is specified, the httpd service can be accessed only through this address.
- To allow multiple ports for access, configure multiple port numbers in **Listen**.
- For details, see section "1.2 Configuring the Apache Home Page" in the lab guide.

Basic Configurations of the Apache HTTP Server – DSO

- Path for storing DSO modules
 - `/usr/lib64/httpd/modules/`
- Query the loaded DSO modules.
 - `httpd -M`
- Load or unload DSO modules.
 - Add or comment out **LoadModule** *Module_name Module path* in the configuration file, for example,
`LoadModule data_module modules/mod_data.so.`

- For details, see section "2.4 Loading and Unloading an Apache DSO Module" in the lab guide.

Basic Configurations of the Apache HTTP Server – MPM

- Default path for storing the MPM configuration file
 - `/etc/httpd/conf.modules.d/00-mpm.conf`
- Configure the working mode of the MPM.
 - `LoadModule mpm_prefork_module modules/mod_mpm_prefork.so`
 - `LoadModule mpm_worker_module modules/mod_mpm_worker.so`
 - `LoadModule mpm_event_module modules/mod_mpm_event.so`
- In openEuler, the default working mode of the MPM is **event**.

- See section "2.5 Changing the Working Mode of the MPM" in the lab guide.

Basic Configurations of the Apache HTTP Server – Persistent Connections

- Common operations:
 - KeepAlive On|Off
 - Enable or disable persistent connection.
 - KeepAliveTimeout 15
 - Set the persistent connection duration to 15 seconds. The default value is **5**.
 - MaxKeepAliveRequests 200
 - Set the maximum number of persistent connection requests. The default value is **100**.

Basic Configurations of the Apache HTTP Server – Static Resource Access

- The Apache HTTP Server allows static resources to be configured as file system paths or URIs.
- A file system path can specify a directory or file on the server, serving as a static resource for users to access.
 - Specify a directory in the configuration file.

```
<Directory "Directory_path">          // Specify the static resource path.  
.....  
</Directory>                         // Grant permissions on the directory.
```

- Specify a file in the configuration file.

```
<File "File_path">  
.....  
</File>
```

- Files can also be represented by wildcards and regular expressions.
- The paths to directories or files can be based on the Apache home directory or redirected to other paths using aliases.

Basic Configurations of the Apache HTTP Server – Static Resource Permissions

- When configuring static resources, the following parameters are used to set permissions:
 - Options
 - **Indexes**: If the home page is not set, the current directory is indexed.
 - **FollowSymLinks**: allows access to source files of the soft link files in the directory.
 - **None**: disables all permissions.
 - **All**: enables all permissions.
 - **AllowOverride**: stores all permission-related configurations in the **.htaccess** file in the specified directory.
 - **All**: enables all directives in the **.htaccess** file.
 - **None**: disables all directives in the **.htaccess** file.
 - **AuthConfig**: enables all directives, except the automatically generated commands.

- If a hyphen (-) is added in front of **Indexes** or **FollowSymLinks**, the function is deleted.

Basic Configurations of the Apache HTTP Server – Static Resource Permissions

- Based on the client address
 - By default, all access requests are denied.
 - **Require all grantd / denied** // *grantd* indicates that all addresses are allowed, and **denied** indicates that all addresses are denied.
 - **Require not ip xx.xx.xx.xx** // **not** must be used together with *grantd* to indicate a blocklist. **denied** is used alone to indicate an allowlist.
- Based on the account
 - There is no account authentication by default.
 - **AuthType** // Authentication type
 - **Basic**: plaintext authentication. *auth_basic_module* must be enabled.
 - **Digest**: digest access authentication
 - **AuthName** // Content in the authentication dialog box
 - **AuthUserFile** // Path to the password file
 - **Require user** // Account that can be used for authentication

- Before modifying account-based permissions, run the **htpasswd** command to create an account and its password.
- It is recommended the password file be stored in the same directory as the accessed static resource.

Basic Configurations of the Apache HTTP Server – Data Compression

- Enable the compression module.

```
LoadModule deflate_module modules/mod_deflate.so SetOutputFilter
```

- Set compression parameters.

```
SetOutputFilter DEFLATE      ##### Options
AddOutputFilterByType DEFLATE text/plain    ##### Specify the type of files to be compressed.
AddOutputFilterByType DEFLATE text/html
AddOutputFilterByType DEFLATE application/xhtml+xml
.....
DeflateCompressionLevel 9      ##### Set the compression level. The value 1 indicates the lowest level, and the value 9 indicates the highest level.
BrowserMatch ^Mozilla/4 gzip-only-text/html      ##### Specify compression or no compression for a specified browser.
BrowserMatch ^Mozilla/4\.0[678] no-gzip
BrowserMatch \bMSI[E] !no-gzip !gzip-only-text/html
```

Basic Configurations of the Apache HTTP Server – Logs

- **access_log** stores access logs of the Apache HTTP Server.
- **error_log** stores error logs of the Apache HTTP Server.
- Log configurations are as follows:

```
<IfModule log_config_module>
    LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\""
    LogFormat "%h %l %u %t \"%r\" %>s %b" common
    CustomLog "logs/access_log" combined
    // Specify the log storage directory and
    logging call format.
</IfModule>
```

Basic Configurations of the Apache HTTP Server – Implementing HTTPS

- Install the dependency.
 - HTTPS depends on the SSL module. mod_ssl must be installed.
- Specify certificate paths.
 - `SSLCertificateFile /path/file` ##### Path to the certificate
 - `SSLCertificateKeyFile /path/file` ##### Path to the private key file
 - `SSLCACertificateFile /path/file` ##### Path to the certificate chain file
- Redirect HTTP to HTTPS through URL redirection.
 - `RewriteEngine on`
 - `RewriteRule ^(/.*)$ https:// %{HTTP_HOST}$1 [redirect=302]`

Basic Configurations of the Apache HTTP Server – Virtual Hosts

- An Apache virtual host runs multiple web services on the same physical server, each with a unique domain name.
- Virtual hosts can be implemented in any of the following ways:
 - Based on different IP addresses
 - Based on different port numbers
 - Based on different domain names
- Virtual host configurations are as follows:

```
<VirtualHost IP:PORT>          // Actual IP address and port number of a virtual host
  ServerName Domain_name        // Domain name of a virtual host
  DocumentRoot "/path"          // Virtual host home page
</VirtualHost>
```

- A server can run multiple virtual hosts. Each virtual host corresponds to a web service, and each web service corresponds to an independent static or dynamic resource. Users can use different domain names, port numbers, or IP addresses to access different resources.

Contents

1. Web Services
2. Apache Overview and Configurations
- 3. NGINX Overview and Configurations**
4. DNS Overview and Configurations
5. MySQL Overview and Configurations
6. LAMP/LNMP

NGINX

- NGINX is a lightweight HTTP and reverse proxy server, a mail proxy server, and a generic TCP/UDP proxy server.
- NGINX can run on multiple platforms, such as x86 and Arm, and supports mainstream OSs, such as Linux and Windows.

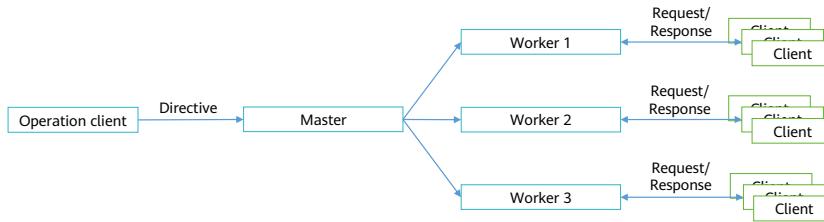
- NGINX is used to set up HTTP and FTP servers, or deployed at the frontend of servers as a reverse proxy and load balancer.

Key Features of NGINX

- High concurrency
 - An NGINX node supports 100,000 concurrent connections, and potentially up to millions of concurrent connections.
- Low memory resource consumption
 - NGINX occupies the least memory of mainstream web servers, consuming only 2.5 MB memory from 10,000 inactive HTTP keep-alive connections.
- High scalability
 - Adopts a modular design that fits many deployments and supports third-party modules.
- High reliability
 - Works in the master-worker mode. If a worker is faulty, the master can quickly start a new worker to provide services.

NGINX Architecture

- Master process
 - Checks whether NGINX is correctly configured.
 - Creates and monitors the number and status of worker processes.
 - Receives management directives from NGINX and performs corresponding operations.
- Worker processes
 - Process client requests.
 - Receive directives from the master process and perform corresponding operations.



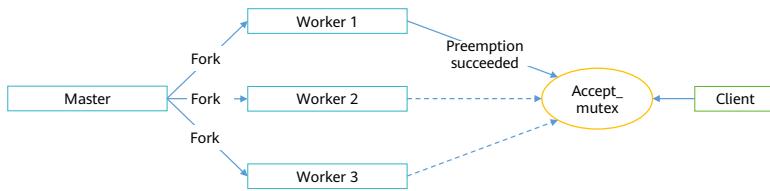
44 Huawei Confidential



- By default, the number of worker processes created by NGINX is the same as the number of CPU cores on the server.
- Worker processes are independent of each other, so a fault does not affect other processes.

Key Working Mechanisms of NGINX

- Preemptive scheduling of worker processes

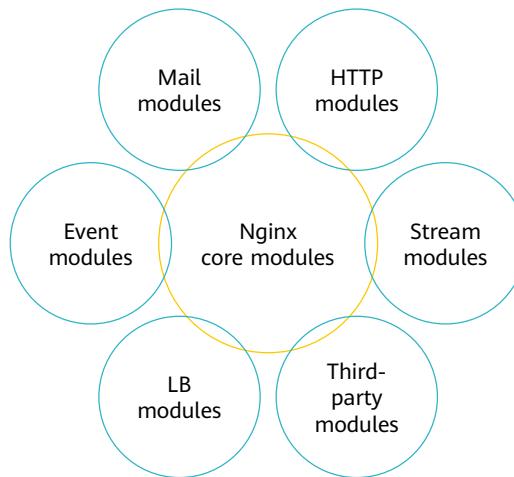


- Asynchronous non-blocking event processing of NGINX



- Preemption: When a client sends a request, the Accept_mutex is activated. The worker that preempts the mutex responds to and processes the request.
- Asynchronous non-blocking event processing: In conventional event processing, if all workers are blocked by the client, new client requests need to wait until the original client releases the workers. In asynchronous non-blocking event processing in which the bottom layer uses the Linux epoll model, even if the workers are blocked, new client requests can also be processed.

Modular System of NGINX



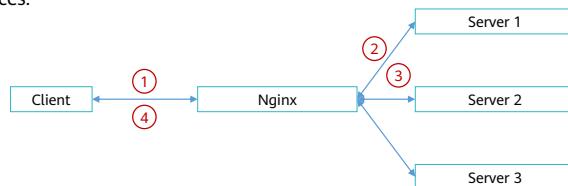
46 Huawei Confidential



- Core modules include `ngx_core_module`, `ngx_errlog_module`, `ngx_events_module`, `ngx_openssl_module`, `ngx_http_module`, and `ngx_mail_module`.
- There are many standardized modules. The modules shown in the figure are only representatives.

NGINX Reverse Proxy

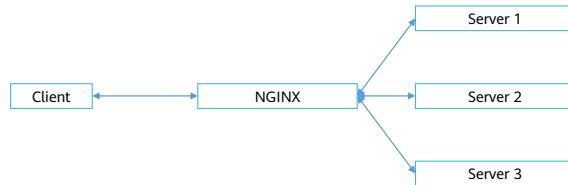
- The working process of the NGINX reverse proxy is as follows:
 1. A client sends a request to NGINX.
 2. After receiving the request from the client, NGINX forwards it to the backend server.
 3. The backend server returns the requested resource to NGINX.
 4. NGINX returns the resource to the client.
- The NGINX reverse proxy is implemented through **proxy_pass** in **location**.
- In this process, the port used by the client to request resources from the proxy can differ from that used by the server to provide services.



- NGINX can serve as a proxy for protocols such as HTTP, HTTPS, TCP, WebSocket, gRPC, POP/IMAP, and RTMP.
- For example, the client uses port 80 to send requests to the proxy, but the server uses port 8080 to provide resources for the reverse proxy.

NGINX Load Balancing

- NGINX provides layer-4 and layer-7 load balancing.
- The load balancing function of NGINX is implemented through upstream in HTTP.
- NGINX provides multiple load balancing algorithms, such as round robin and weight.
- When the round robin algorithm is used, NGINX sends requests to different servers in turn so that each server provides services equally.
- When the weight algorithm is used, NGINX sends requests to servers based on the configured weights. Servers with good performance provide more services, and servers with poor performance provide fewer services.



Automatically Installing NGINX Using Yum

- Installed using Yum

- **yum install -y nginx**

```
[root@localhost yum.repos.d]# yum install -y nginx
Last metadata expiration check: 0:00:23 ago on Thu 2018-01-18 09:23:24AM CST.
Dependencies resolved.
=====
Package                                         Architecture
=====
Installing:
nginx                                         x86_64
Installing dependencies:
gd                                              x86_64
gperf-tools-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
=====
Transaction Summary
=====
Install 14 Packages
Total download size: 1.6 M
Total installed size: 5.3 M
```

Compiling and Installing NGINX

- Download the source package from the following path:
 - <https://nginx.org/en/download.html>
- Install the environment dependencies.
 - `yum install -y gcc gcc-c++ make pcre pcre-devel zlib-devel openssl openssl-devel`
- Install NGINX by compiling source code.

- Create the makefile.

```
./configure \
--prefix=/usr/local/nginx \
--pid-path=/var/run/nginx/nginx.pid \
.....
```

- Install NGINX by compiling source code:

```
make && make install
```

- Installation by compiling source code has the following advantages:
 - You can select a specific version.
 - You can customize the installation path, modules, and third-party plug-ins.
- OpenSSL is used for HTTPS services.
- During the compilation, an error indicating that specific dependencies are missing may be reported. In this case, you need to install the dependencies as prompted.
- For details about the parameters added when the makefile is created and their functions, see the official document at <https://nginx.org/en/docs/configure.html>.

Common NGINX Commands

- Check the NGINX configuration file: **nginx -t**
- Replace the default configuration file with a specified configuration file: **nginx -c file**
- View the NGINX version: **nginx -v**
- View detailed information such as the NGINX version, compiler version, and configuration parameters: **nginx -V**
- Start NGINX: **nginx**
- Forcibly stop NGINX: **nginx -s stop**
- Gracefully stop NGINX: **nginx -s quit**
- Restart NGINX: **nginx restart**
- Reload NGINX: **nginx -s reload**

NGINX Service Files

- NGINX configuration files are stored in **/etc/NGINX**.
 - **/etc/nginx/conf.d** is the sub-directory for storing configuration files.
 - **/etc/nginx/nginx.conf** is the main configuration file.
 - **/etc/nginx/fastcgi_params** is used to translate NGINX variables for PHP to identify.
 - **/etc/nginx/mime.types** is used to configure the supported media file types.
 - **/etc/nginx/uwsgi_params** is used to translate NGINX variables for Python to identify.
 - **/etc/share/nginx/html** is the default root directory of the NGINX website.
 - **/etc/log/nginx** is the default NGINX log directory.

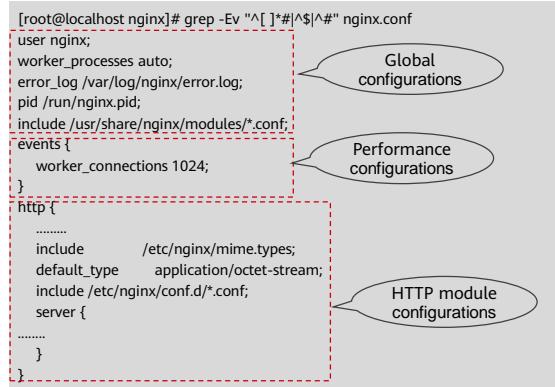
```
[root@localhost nginx]# rpm -ql nginx
/etc/ima/digest_lists.tlv/0-metadata_list-compact_tlv-nginx-1.21.5-3.oe2203.x86_64
/etc/ima/digest_lists/0-metadata_list-compact-nginx-1.21.5-3.oe2203.x86_64
/etc/nginx
/etc/nginx/fastcgi.conf
/etc/nginx/fastcgi_params
/etc/nginx/fastcgi_params.default
/etc/nginx/koi-win
/etc/nginx/mime.types
/etc/nginx/mime.types.default
/etc/nginx/nginx.conf
/etc/nginx/nginx.conf.default
/etc/nginx/scgi_params
/etc/nginx/scgi_params.default
/etc/nginx/uwsgi_params
/etc/nginx/uwsgi_params.default
/etc/nginx/win-utf
/usr/bin/nginx-upgrade
/usr/lib/systemd/system/nginx.service
/usr/lib64/nginx/modules
/usr/sbin/nginx
/usr/share/licenses/nginx
/usr/share/licenses/nginx/LICENSE
/usr/share/nginx/html/404.html
/usr/share/nginx/html/50x.html
/usr/share/nginx/html/index.html
/usr/share/nginx/html/nginx-logo.png
/usr/share/vim/vimfiles/ftdetect/nginx.vim
/usr/share/vim/vimfiles/indent/nginx.vim
/usr/share/vim/vimfiles/syntax/nginx.vim
/var/lib/nginx
/var/lib/nginx/tmp
/var/log/nginx
```

Structure of the Main Configuration File nginx.conf of the NGINX Service

- Global configurations
 - Running user
 - Number of workers
 - Log parameters
 - External configuration files imported by **include**
- Performance configurations
 - Working mode
 - Number of TCP connections
- HTTP module configurations
 - HTTP request and response parameters
 - Route
 - Virtual host
 - Reverse proxy
 - Load balancing

```
[root@localhost nginx]# grep -Ev "[ ]#|^$|^#" nginx.conf
user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log;
pid /run/nginx.pid;
include /usr/share/nginx/modules/*.conf;
events {
    worker_connections 1024;
}

http {
    .....
    include      /etc/nginx/mime.types;
    default_type application/octet-stream;
    include /etc/nginx/conf.d/*.conf;
    server {
        ...
    }
}
```



- Except the global configurations, other configurations in the NGINX configuration files are marked with braces ({}). The configurations are separated as modules.

Main Configuration File of the NGINX Service – Global Configurations

- The user of the master process is **root**, while the user of worker processes can be changed by **user** in the global configurations.
- By default, the number of worker processes is the same as that of CPU cores on the server. Configure **worker_processes** in the global configurations to change this limit.

```
user nginx;           ##### User to which a worker process belongs
worker_processes auto; ##### Number of worker processes
error_log /var/log/nginx/error.log; ##### Path for storing error logs
pid /run/nginx.pid; ##### Path to the process file
include /usr/share/nginx/modules/*.conf; ##### Path of a functional module
```

Main Configuration File of the NGINX Service – HTTP Configurations

```
http {
    log_format main '$remote_addr - $remote_user [$time_local] "$request"
                    '$status $body_bytes_sent "$http_referer"
                    '"$http_user_agent" "$http_x_forwarded_for"';
    access_log /var/log/nginx/access.log main;
    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    packets keepalive_timeout 65;
    types_hash_max_size 4096;
    include /etc/nginx/mime.types;
    default_type application/octet-stream;
    include /etc/nginx/conf.d/*.conf;
    server {
        listen 80;
        listen [::]:80;
        server_name _;
        root /usr/share/nginx/html;
        include /etc/nginx/default.d/*.conf;
        error_page 404 /404.html;
        location = /404.html {
        }
        error_page 500 502 503 504 /50x.html;
        location = /50x.html {
    }
}
```

55 Huawei Confidential



- **tcp_nopush** is an optimization parameter used to send data packets immediately. **on** indicates do not send, and **off** indicates send. If packets are sent immediately, congestion may occur. You are advised to set this parameter to **on**.
- If **tcp_nodelay** is set to **off**, the communication latency is increased, but the bandwidth utilization is improved. In high-latency scenarios with a large amount of data, **tcp_nodelay on** will increase the number of small packets but delay the response. In latency-sensitive scenarios, this setting has a good effect.
- **types_hash_max_size** affects the conflict rate of the hash list. If **types_hash_max_size** is set to a large value, more memory will be consumed, but the conflict rate of hash keys will decrease and retrieval is faster. Otherwise, a small value will use less memory, but result in more conflicts of hash keys.

Main Configuration File of the NGINX Service – HTTPS Configurations

```
server {  
    listen      443 ssl http2;                      ##### Listening IPv4 port  
    listen      [::]:443 ssl http2;                  ##### Listening IPv6 port  
    server_name _;  
    root        /usr/share/nginx/html;  
  
    ssl_certificate "/etc/pki/nginx/server.crt";  
    ssl_certificate_key "/etc/pki/nginx/private/server.key";  
    ssl_session_cache shared:SSL1m;  
    ssl_session_timeout 10m;  
    ssl_ciphers PROFILE=SYSTEM;  
    ssl_prefer_server_ciphers on;  
  
    include /etc/nginx/default.d/*.conf;  
  
    error_page 404 /404.html;  
        location = /40x.html {  
    }  
  
    error_page 500 502 503 504 /50x.html;  
        location = /50x.html {  
    }  
}
```

Main Configuration File of the NGINX Service – Transmission Compression

- The transmission compression configurations are in the **http** module.
- A higher compression ratio will result in more CPU performance loss.

```
http {  
....  
    gzip on;                      ##### Enable gzip compression.  
    gzip_min_lenth 1;              ##### Minimum size for compression. If this parameter is set to 1, files smaller  
than 1 byte will not be compressed.  
    gzip_comp_level 3;             ##### Compression ratio. The maximum value is 9 and the minimum value is 1.  
    gzip_types text_plain application/javascript ##### Types of files that can be compressed  
    server {  
....  
        }  
    }  
}
```

Main Configuration File of the NGINX Service – Virtual Hosts

- NGINX uses the **server** directive to configure virtual hosts in the HTTP module.
- Virtual hosts can be implemented in the following three ways in NGINX:
 - Based on different IP addresses
 - Based on different port numbers
 - Based on different domain names
- Syntax of the **server** directive

```
http {  
    server {  
        # Server configuration  
    }  
}
```

Main Configuration File of the NGINX Service – Routing Rule

- NGINX uses the **location** directive located in the virtual host configuration module for routing.
- Syntax of the **location** directive: **location [=|~|~*|^~] /uri/{ ... }**
 - = indicates exact match.
 - ^~ indicates that the URI starts with a common character string and does not match any regular expression.
 - ~ indicates case-sensitive regular expression match.
 - ~* indicates case-insensitive regular expression match.
 - / indicates general match. If there is no other match, any requests will be matched.
- If the URI is a directory, it must end with a slash (/).
- In **location**, you can specify a path in either of the following ways:
 - **root**: relative path
 - **alias**: absolute path
- The autoindex function can be enabled in the **location** configuration module, indicating that the accessed directory can be indexed.

```
http {
.....
server {
.....
    location / {
        root /data;
        index index.html;
    }
}
}
```

- Priorities of **location** expressions: **location = > location Full_path > location ^~ Path > location , * Regular_order > location Partial_start_path > /**

NGINX Hot Upgrade

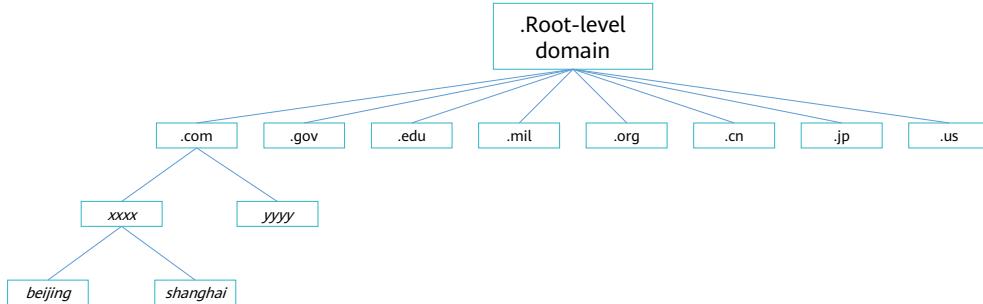
- NGINX hot upgrade refers to upgrading NGINX of a source version to a specified version without affecting client requests.
- The NGINX hot upgrade process is as follows:
 - Obtain the new NGINX binary file through compilation.
 - Back up the NGINX file of the source version.
 - Replace the NGINX file of the source version with that of the target version.
 - Send the USR2 signal to the master process of the source version.
 - Start the master process of the target version.
 - Verify that NGINX of the target version works properly.
 - Exit the NGINX process of the source version.

Contents

1. Web Services
2. Apache Overview and Configurations
3. NGINX Overview and Configurations
- 4. DNS Overview and Configurations**
5. MySQL Overview and Configurations
6. LAMP/LNMP

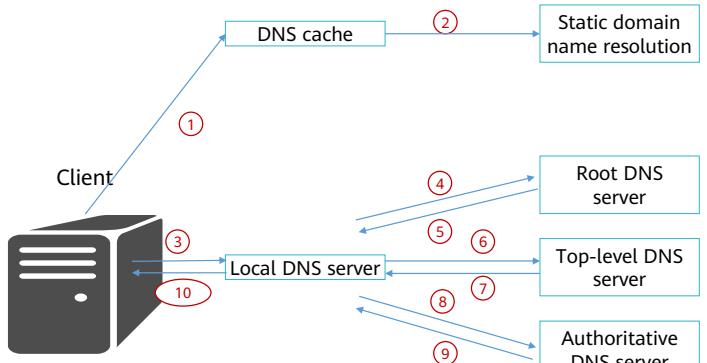
DNS

- Domain Name System (DNS) is an Internet service that translates domain names to server IP addresses, streamlining Internet access and reducing O&M workloads.
- As an application-layer protocol, it uses both UDP and TCP port 53 and works based on the client-server model.
- In Linux, the DNS service is usually implemented using Berkeley Internet Name Domain (BIND).
- DNS is hierarchical, with the root at the top and various levels of domains, subdomains, and records below. A maximum of 127 levels of subdivisions are supported.



Forward DNS Resolution

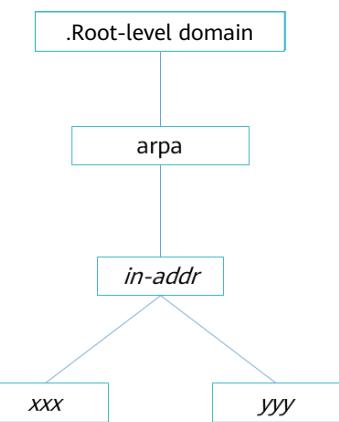
1. The OS checks whether the corresponding IP address exists in the local cache.
2. The OS checks whether static domain name resolution is configured in the local `hosts` file.
3. The OS sends a query request to the local DNS server.
4. If the local DNS server does not have the requested domain name, it sends a query request to the root server.
5. The root server provides the IP address of the top-level DNS server for the local DNS server.
6. The local DNS server sends a resolution request to the top-level DNS server.
7. The top-level DNS server returns the IP address of the authoritative DNS server to the local DNS server.
8. The local DNS server sends a resolution request to the authoritative DNS server.
9. The authoritative DNS server completes address resolution and returns the address corresponding to the domain name to the local DNS server.
10. The local DNS server returns the address to the client.



- The preceding figure shows an iterative DNS query process. DNS also supports recursive query.
- In a recursive query, the IP address of the top-level DNS server is not directly returned to the local DNS server in step 5, and instead the root directly sends a resolution request to the top-level DNS server. In addition, the IP address of the authoritative DNS server in step 7 is not returned to the local DNS server. Instead, the root continues to send a resolution request to the authoritative DNS server, and the authoritative DNS returns the resolved address to the top-level DNS and then to the root DNS. The root DNS then returns the resolved address to the local DNS server.

Reverse DNS Resolution

- Reverse DNS resolution is to resolve IP addresses into domain names.
- It is mainly used for mail domain name resolution.
- **arpa** is the top-level domain name in reverse DNS resolution.



DNS Resource Records

- Resource records are used by the DNS server to resolve domain names. They are equivalent to DNS database files.
- Common resource types include A, AAAA, SOA, NS, PTR, CNAME, and MX.
- Resource records are in a fixed format, that is, *name [TTL] IN RR_TYPE VALUE*.
 - *name*: domain name.
 - *TTL*: time to live, which can be inherited from the global configuration.
 - *IN*: Internet record. The format is fixed.
 - *RR_TYPE*: type of a resource record.
 - *VALUE*: value. The value varies according to resource records.

DNS Resource Record Type – SOA

- There must be exactly one start of authority (SOA) record per zone.
- The SOA record starts every zone file.
- *name* in an SOA record is the name of the domain.
- *VALUE* in an SOA record consists of many parts.
- An example SOA record is shown on the right, where:
 - **\$TTL** globally defines the time to live for the domain. This field can be left blank for other resource records.
 - **@** specifies the name of the domain in the resource record.
 - **master.test.com** indicates the master DNS server.
 - **xxx.test.com** indicates the email address of the administrator. Because the at sign (@) has a special meaning in the SOA record, a period (.) is used to replace the at sign (@) in the email address.
 - **0** indicates the version number, which is used by the slave DNS server to determine whether to synchronize information.
 - **1D** indicates the interval for synchronizing data from the slave server to the master server.
 - **1H** indicates the retry interval for synchronizing data from the slave server to the master server.
 - **1W** indicates the period during which the slave server determines whether the master server becomes invalid. For example, **1W** indicates that the master server becomes invalid if it fails to synchronize data with the slave server within one week.
 - **3H** indicates the time to live for invalid domain names. Within this period of time, if users query any invalid domain names again, the server directly returns a response indicating that the domain names cannot be resolved and does not query the domain names again.

```
$TTL 1D
@ IN SOA master.test.com. xxx.test.com. (
          0      ; serial
          1D     ; refresh
          1H     ; retry
          1W     ; expire
          3H )   ; minimum
```

- In an SOA record, you can use an at sign (@) to replace a domain name, otherwise, a domain name must end with a period (.). For example, if the domain name is **test.com**, enter **test.com.** in the SOA record.
- A record of type A needs to be set for **master** in **master.test.com** to resolve the domain name into an IP address.

Common Types of DNS Resource Records

- A is used to resolve a domain name into an IPv4 address, and AAAA is used to resolve a domain name into an IPv6 address. In an A resource record, *name* indicates the FQDN of the host, and *VALUE* indicates the IP address of the host.
- NS indicates a name server record. A zone can contain multiple NSs. Among these NSs, one is the master DNS server, and the others are slave DNS servers.
- MX identifies the IP address of the mail server in a zone. A zone can contain multiple MXs. MX uses 0 to 99 to indicate the priority. The value **0** indicates the highest priority and the value **99** indicates the lowest priority.
- CNAME indicates the alias of a domain name.
- PTR is used for reverse resolution that resolves domain names into IP addresses. PTR records are unique. They start with **.arpa** and end with the host bit of the address.

- In MX, *name* is the domain name, and *VALUE* is the host name of the mail server.
- In CNAME, *name* is the FQND of the alias, and *VALUE* is the actual FQDN.
- In an A record, a host name can point to multiple IP addresses. An asterisk (*) can be specified to resolve domain names of hosts that are not defined into specified IP addresses.

Key Configuration Files of the DNS Service

```
[root@localhost ~]# rpm -ql bind
...
/etc/logrotate.d/named          // Log rotation file
/etc/named                       // Main directory of configuration files
/etc/named.conf                   // Main configuration file
/etc/named.rfc1912.zones         // Definition zone
/usr/sbin/named                  // Binary command
/usr/sbin/named-checkconf        // Command for checking the configuration file
/usr/sbin/named-checkzone        // Command for checking the zone file
/var/log/named.log               // Log file
/var/named                        // Main directory of data files
/var/named/named.ca              // Root domain server
/var/named/named.localhost        // Template of the forward resolution zone file
/var/named/named.loopback         // Template of the reverse resolution zone file
/var/named/slaves                 // Default path for downloading files from the DNS
server
....
```

Common DNS Commands

- Run the **named-checkconf** command to check whether the DNS configuration has syntax errors.

```
[root@localhost ~]# named-checkconf  
/etc/named.conf:19: missing ';' before '}'
```

- Run the **named-checkzone** *Domain_name Data_file* command to check whether a DNS data file contains syntax errors.

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

- Run the **rndc reload** command to reload DNS configurations.

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

Common DNS Utilities – nslookup

- The **nslookup** command is used to check whether domain name resolution is normal and diagnose network faults.
 - Interactive: used for multiple queries and diagnosis
 - Non-interactive: used for single query and diagnosis

```
[root@localhost ~]# nslookup  
> www.test.com  
Server: 192.168.38.159  
Address: 192.168.38.159#53  
  
www.test.com canonical name = main.test.com.  
Name: main.test.com  
Address: 192.168.38.157
```

```
[root@localhost ~]# nslookup www.test.com  
Server: 192.168.38.159  
Address: 192.168.38.159#53  
  
www.test.com canonical name = main.test.com.  
Name: main.test.com  
Address: 192.168.38.157
```


Common DNS Utilities – host

- The **host** command is a simple utility for performing DNS lookups, in which it queries the forward and reverse DNS resolution results.

- Common options of the **host** command:

- a**: displays detailed DNS information.
- c**: specifies the query type.
- C**: queries the complete SOA record of a specified host.
- t**: specifies the type of the domain name to be queried.
- 4**: uses IPv4 for query.
- 6**: uses IPv6 for query.

```
[root@localhost ~]# host -a www.test.com
Trying "www.test.com"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id:
24111
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0,
ADDITIONAL: 0
;; QUESTION SECTION:
;www.test.com.           IN      ANY
;; ANSWER SECTION:
www.test.com.      86400   IN      CNAME  main.test.com.
Received 49 bytes from 192.168.38.159#53 in 1 ms
[root@localhost ~]# host -C test.com
Nameserver 192.168.38.159:
test.com has SOA record master.test.com.
admin.test.com. 0 86400 3600 604800 10800
```

Contents

1. Web Services
2. Apache Overview and Configurations
3. NGINX Overview and Configurations
4. DNS Overview and Configurations
- 5. MySQL Overview and Configurations**
6. LAMP/LNMP

What Is Data?

- Data is the representation of information in the form of symbols, text, numbers, voices, images, or videos. Data is basically a form of information: data itself is meaningless and becomes information only when it is put to use.
- Data in a computer system is expressed as various letters, numbers, voices, symbols, images, and more, which are formed through binary bits 1 and 0. Binary is converted into more conventional forms of data through processing.



HUAWEI

- Symbols that represent things are called data. Data may be represented as anything, be it a number, text, symbol, image, voice, or language. Data is recorded in a database. For example, a student management database would include information about a student, including the student's number, name, gender, age, date of birth, and phone number.
- Information that can be read and used in society and production is extracted from data through processing. Unprocessed data is akin to a raw material, and its value lies in its ability to record facts about the real world.

Database

- A database is an, often large, collection of organized data that is stored in a computer, often for a long time, and with the ability to be accessed. Data in a database is organized, described, and stored according to a data model. The data in a database is efficient, relatively independent, and scalable, and can be accessed by various users.



- A database is a collection of organized and accessible data stored in a computer, generally for a long time. Generally speaking, a database is a place to store data, just as a refrigerator is to store food. You might not know it, but everyone is using databases. When we look up names in the phone book, we're using a database. When searching the Internet, we are also using a database. When we log in to the network, we need to verify our name and password in the database. Even when we use an ATM, a database is used to verify the PIN code and check your balance.
- A database is, in essence, a set of files or a file system, that is, a repository that stores data in a specific format. Users can add, delete, modify, and query the stored data.
- In daily life, people use languages and their voice to describe things. In a computer, features of these things are broken down and recorded.
- A database is an organized collection of structured information or data, typically stored electronically in a computer system.
- A database can be regarded as an electronic file cabinet, where electronic files are stored. Users can add, truncate, update, and delete data in files.
- A database is a collection of data that is stored together in a certain way. It can be accessed by multiple users, usually has minimal redundancy, and operates independently of applications.
- Image source: <http://3ms.huawei.com/km/static/image/detail.html?fid=62237>

DBMS

- A database management system (DBMS) consists of a collection of interconnected data and a set of programs used to access the data. This data set is usually called a database. The main purpose of DBMS is to provide a convenient and efficient way to access database information.
- A DBMS acts as an interface between a database and its users or programs, allowing users to retrieve, update, and manage information in an organized and optimized manner. In addition, the DBMS helps monitor and control the database and provides various management operations, such as performance monitoring, optimization, backup, and restoration.
- Typical DBMSs include Oracle, Microsoft SQL Server, Access, MySQL, and PostgreSQL.

- A database is usually inseparable from its database software program, that is, a DBMS. A DBMS acts as an interface between a database and users or programs, allowing users to retrieve, update, and manage information in an organized and optimized manner. It helps monitor and control the database and provides various management operations, such as performance monitoring, optimization, backup, and restoration.
- Database software is used to create, edit, and maintain database files and records, helping perform common operations such as file and record creation, data entry, data editing, and update. In addition, database software is used for data storage, backup, reporting, multi-channel access control, and security issues. With the increasing occurrence of data theft today, database security has become essential. Database software is sometimes called a DBMS.
- Common database software includes MySQL, Microsoft Access, Microsoft SQL Server, FileMaker Pro, Oracle Database, and dBase.

DBMS Ranking

| Rank | | | DBMS | Database Model | Score | | |
|-------------|-------------|-------------|--|--|-------------|-------------|-------------|
| Feb 2023 | Jan 2023 | Feb 2022 | | | Feb 2023 | Jan 2023 | Feb 2022 |
| 1. | 1. | 1. | Oracle  | Relational, Multi-model  | 1247.52 | +2.35 | -9.31 |
| 2. | 2. | 2. | MySQL  | Relational, Multi-model  | 1195.45 | -16.51 | -19.23 |
| 3. | 3. | 3. | Microsoft SQL Server  | Relational, Multi-model  | 929.09 | +9.70 | -19.96 |
| 4. | 4. | 4. | PostgreSQL  | Relational, Multi-model  | 616.50 | +1.65 | +7.12 |
| 5. | 5. | 5. | MongoDB  | Document, Multi-model  | 452.77 | -2.42 | -35.88 |
| 6. | 6. | 6. | Redis  | Key-value, Multi-model  | 173.83 | -3.72 | -1.96 |
| 7. | 7. | 7. | IBM Db2 | Relational, Multi-model  | 142.97 | -0.60 | -19.91 |
| 8. | 8. | 8. | Elasticsearch | Search engine, Multi-model  | 138.60 | -2.56 | -23.70 |
| 9. | ↑ 10. | ↑ 10. | SQLite  | Relational | 132.67 | +1.17 | +4.30 |
| 10. | ↓ 9. | ↓ 9. | Microsoft Access | Relational | 131.03 | -2.33 | -0.23 |
| 11. | ↑ 12. | 11. | Cassandra  | Wide column | 116.22 | -0.09 | -7.76 |
| 12. | ↓ 11. | ↑ 15. | Snowflake  | Relational | 115.65 | -1.60 | +32.47 |
| 13. | 13. | ↓ 12. | MariaDB  | Relational, Multi-model  | 96.81 | -2.55 | -10.30 |

- Data source: <https://db-engines.com/en/ranking>

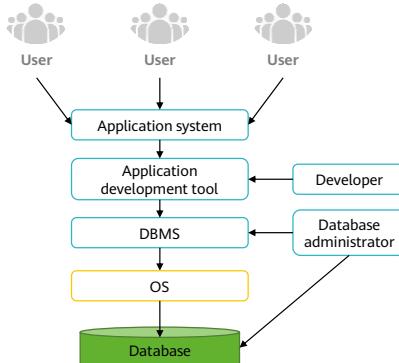
DBMS – Functions

- **Data definition:** The DBMS provides a data definition language (DDL), through which users can easily define data objects in a database.
- **Data manipulation:** The DBMS also provides a data manipulation language (DML), which can be used by users to manipulate data and perform basic operations on the database, such as query, insert, delete, and modify.
- **Database management:** The DBMS manages and controls the creation and O&M of databases to ensure data security, integrity, and concurrency, and ensures the system recovers after a fault.
- **Interfaces and tools for convenient and effective database access:** Programmers can develop database applications through an interface. Database administrators (DBAs) can use tools to manage databases.
- **Database creation and maintenance:** include the input and conversion of initial data to the database, the dump and restoration of the database, the reorganization of the database, and the performance monitoring and analysis. These functions are usually implemented in software.

- **Database management:** The DBMS manages and controls the creation and O&M of databases to ensure data security, integrity, and concurrency, and ensures the system recovers after a fault. For example:
 - Data integrity checks ensure data meets the requirements of the database.
 - Security protection ensures only authorized users can access data.
 - Concurrency means multiple users can access data at the same time.
 - Fault rectification restores the database for users in the event there is a fault, providing reliable access to the database.

DBS

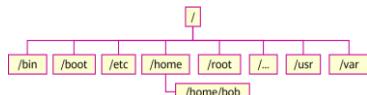
- A database system (DBS) is the sum of all the database components, like the database, DBMS, application development tool, application system, database administrator, and users.



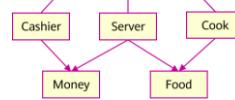
- A DBS consists of the following three parts:
 - Database: stores data.
 - DBMS: manages the database.
 - Database applications: supplement the DBMS and improve processing capabilities of the DBS.
- A database (DB for short) provides space for storing various data, and can be considered as a container. A database may contain many files, and a DBS usually contains many databases.
- A DBMS is the software used by users to create, manage, and maintain a database. It serves between users and the OS to manage the database in a unified manner. The DBMS defines the data storage structure, provides the mechanism for data operations, and maintains the security, integrity, and reliability of the database. However, it cannot meet the requirements for data management in many cases.
- Database applications meet data management requirements and streamline the management process. Database applications communicate with the DBMS, access and manage data stored in the DBMS, and allow users to insert, modify, and delete data in the database.

Database Models

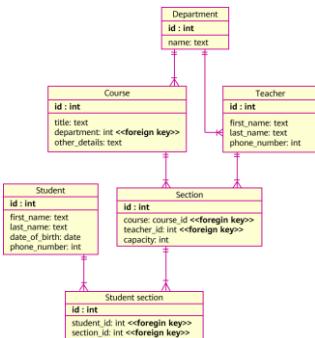
- Since the middle of the 20th century, many interesting database models, such as **hierarchical model**, **network model**, and **relational model**, have emerged. Some database models are no longer used, while others are still playing a significant role.



Hierarchical model



Network model

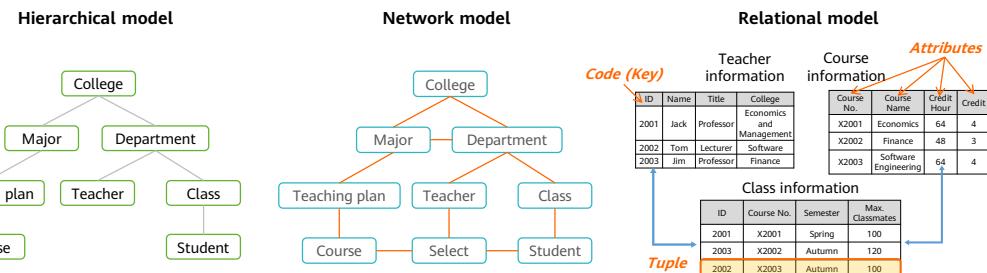


Relational model

- In the hierarchical model, a tree structure is used to indicate the relationship between data records. In the network model, a graph structure is used to indicate the relationship between data records. In the relational model, a two-dimensional table (or relationship) is used to indicate the relationship between data records.
- Hierarchical database
- The hierarchical model came about in 1960s. It came about as people tried to process more complex data. Similar to a tree structure, each record has only one parent node. It improves data reads using specific methods but access to this type of model is similar to traversing a linked list. So it struggles to read complex data in most situations.
- Although hierarchical databases have little value when applied to modern systems, they are still used in Linux/Windows file systems, DNS, and LDAP systems. Database implementations:
 - Linux/Windows file system
 - DNS
 - LDAP

- Network database
- Network databases were also created in the 1960s. The Committee of the Data Systems Language Conference (CODASYL) standardized the network model. As such, the network model is also referred to as the CODASYL model.
- The network database further develops the hierarchical model. Developments mean a child node can have more than one parent node, increasing its complexity. Similar to the hierarchical database, query and updates to the network database require traversing a linked list, bringing endless difficulty to application development and making the network database eventually discarded. Now, there are better alternatives. Database implementation:
 - IDMS
- Relational database
- The relational model was born in 1960s. It is the most active, lively, and widely used data model in the existing database models. The relational model was not popular at the very beginning, but became the first choice of most in the 1980s. The relational model is based on relational algebra, where data can be organized into relationships (called tables in SQL) and each relationship is an unordered set of tuples (called rows in SQL). In other words, a relationship (table) is only a collection of tuples (rows). The relational model solves the age-old problem of expressing many-to-many relationships. In a relational database, developers can read any rows and columns in a table without traversing a linked list. In addition, foreign keys are used to associate tables. SQL is a language used in relational databases for querying data. As a Turing-complete language, SQL is not limited to relational databases, but has taken a dominant position in the database field. The core of SQL is the query optimizer, which automatically determines the SQL execution sequence and indexes used. In general, the relational model has become the first choice of most developers because of its strong flexibility and adaptability. Database implementations:
 - MySQL
 - PostgreSQL
 - SQLite

Database Models – Hierarchical, Network, and Relational Models



82 Huawei Confidential



- Hierarchical database
- The hierarchical database was the first successful database system to be developed. It presents data in a hierarchical structure (tree structure). The hierarchical databases used to be the mainstream of databases, but with the appearance and popularization of the relational databases, they have become obsolete. A typical hierarchical database is the Information Management System (IMS) database, which was developed by IBM.
- Relational database
- The relational database is the most widely used database. It was born in 1969 and has a long history. Like an Excel worksheet, a relational database uses a two-dimensional table consisting of rows and columns to manage data, making it easy to understand and easy to read and query. In addition, SQL is used to perform operations on relational database data.

- [File system stage]
- Since the mid-1960s, direct access storage devices such as disks and magnetic drums have been moved to the inside of the computer, where data is stored in files and managed by the file system. The file system allows access to the data in files based on a file storage path and file name. You can view, modify, add, and delete these files. Compared with manual management, file systems are streamlined, meaning you do not need to search for a file by hand. However, the data in these files is not managed in a structured manner, and it is inconvenient to query for the data within.
- The features of data management with a file system are as follows:
 - Data can be stored for a long time.
 - Data is managed by the file system.
 - High data independency.
 - In addition to poor data redundancy and concurrency, it is unable to cope with emergencies (for example, files deleted by mistake or faulty disks).
- [DBS stage]
- In the late 1960s, with the development of network technologies and the improvement of computer software and hardware, database technology emerged. This stage is the so-called DBS stage. At the DBS stage, a dedicated database is used to manage data. You can create a database in the DBS, create tables in the database, and store data in these tables. You can query the DBMS for data in a table. The DBS structures the data. In a file system, data within a file can be structured, but there are no relationships established between files, meaning there is no overarching structure. Although the DBS is often divided into many separate data files, it pays more attention to the relationship between the data in the same database.
- Data management at the DBS stage has the following features:
 - Unified data management.
 - High access concurrency and redundancy.
 - Data is independent.
 - Fine data granularity.
 - Data granularity defines how detailed data in the database is. A finer granularity indicates more details, and a coarser granularity indicates fewer details.

Basic Concepts of Relational Databases

- Data in a relational database is stored as relationships, a two-dimensional table consisting of tuples (records) and attributes (fields). Relationships are presented as a data table to users.
- Each row in a data table is called a record.
- Each column of a data table is called an attribute.
- The primary key in the data table is an attribute that identifies a record, making the record unique globally.
- A field in the data table specifies the value range of an attribute.

- Theoretically, there can be an unlimited number of rows and columns in a data table, but note many rows and columns will cause performance deterioration.
- Primary key

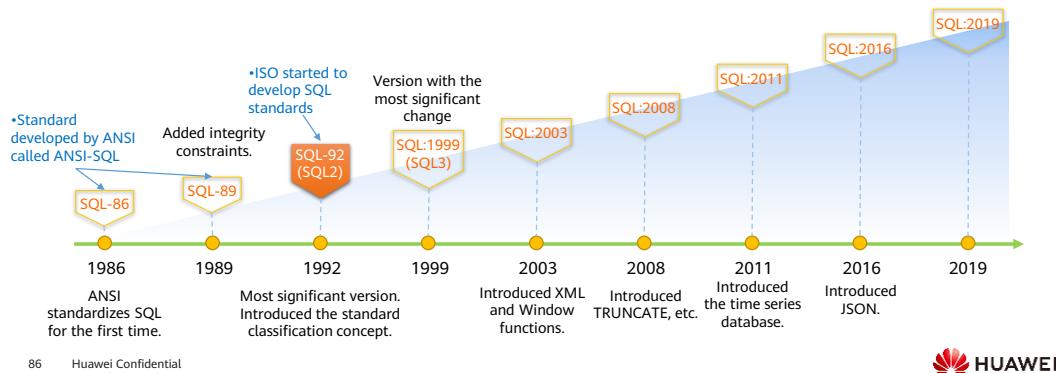
Data

- Structured data
 - Fixed format and limited length.
- Unstructured data
 - No fixed format or length.
- Semi-structured data
 - Combines structured and unstructured data. It is self-describing and does not conform to a data model but has some structure.

- Typical structured data is like the nationality, gender, and age of a person.
- Unstructured data is dominating today's data volume, especially voice and video data.
- Common semi-structured data includes XML and HTML files.

SQL

- Structured Query Language (SQL) is the standard language for relational databases. It is a general-purpose and powerful relational database language. It is a standard interface for accessing relational data and is also the basis for interoperability between different DBSs.
- SQL is based on relational algebra and tuple relationship calculations. It integrates data query, data operations, data definition, and data control functions. The scope of SQL includes data insertion, query, update, and deletion, database schema creation and modification, and data access control.



86 Huawei Confidential



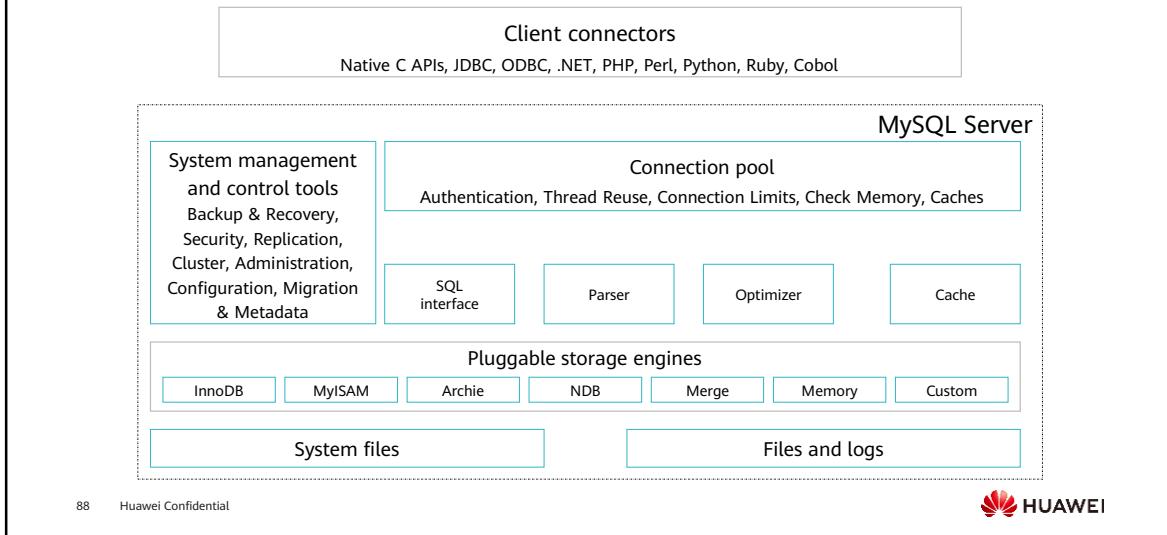
- SQL is a widespread language for relational databases. It is supported by large-scale commercial database products such as Oracle, DB2, Sybase, and SQL Server, and open-source databases like PostgreSQL and MySQL, and even products like Access. In recent years, NoSQL system once claimed that SQL was no longer needed; however, it had to amend it to "Not Only SQL" later to emphasize its coexistence with SQL.
- 1986: ANSI X3.135-1986, ISO/IEC 9075:1986, SQL-86 is the first time that ANSI has standardized SQL. 1989: ANSI X3.135-1989, ISO/IEC 9075:1989, SQL-89 added integrity constraints. 1992: ANSI X3.135-1992, ISO/IEC 9075:1992, SQL-92 (SQL2) is the most significant version, and the standard classification concept is introduced. 1999: ISO/IEC 9075:1999, SQL:1999 (SQL3) is the version with the most significant change; it changed the definition of standards compliance and added support for object-oriented features, regular expressions, stored procedures, and Java. 2003: ISO/IEC 9075:2003, SQL:2003 introduced XML and Window functions. 2008: ISO/IEC 9075:2008, SQL:2008 introduced TRUNCATE. 2011: ISO/IEC 9075:2011, SQL:2011 introduced the time series database. 2016: ISO/IEC 9075:2016, SQL:2016 introduced JSON.

MySQL

- MySQL is the most popular open source SQL DBMS developed, released, and supported by Oracle:
 - MySQL is a DBMS.
 - MySQL databases are relational.
 - MySQL software is open source.
 - The MySQL Database Server is fast, highly available, scalable, and easy to use.
 - The MySQL Server works in client/server or embedded systems.
 - A large amount of contributed MySQL software is available.
- MySQL official website: <https://www.mysql.com/>

- Content source: <https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html>

MySQL Architecture



- Connectors: MySQL Connectors provide connectivity to the MySQL server for client programs.
- Connection pool: Connection pooling is a technique to create and manage a pool of connections that are ready for use by any thread.
- System management and control tools: provide various database management functions, such as backup and recovery, and database and table division.
- SQL interface: receives SQL commands from users and returns query results.
- Parser: parses and verifies SQL statements.
- Optimizer: optimizes SQL statements.
- Cache: If a query result is hit in the query cache, the query statement can directly obtain data from the query cache without executing the SQL statement.
- Pluggable storage engines: MySQL Server uses a pluggable storage engine architecture that enables storage engines to be loaded into and unloaded from a running MySQL server. The storage engines are based on tables not databases.
- System files, files and logs: physical files, including redolog, undolog, binlog, errorlog, querylog, slowlog, data, and index files.

Installing MySQL

Install MySQL in any of the following ways:

- Use the Yum repository provided by openEuler for installation.
- Download the Yum repository configuration tool of the required version from the MySQL official website. After the configuration is complete, use the Yum repository to install MySQL.
- Download the RPM package of the required version from the MySQL official website and run the **rpm** command to install MySQL.
- Download the required source package from the MySQL official website, and compile source code for installation.

- In this course, the Yum repository provided by openEuler is used to install MySQL 8.0.x. The Yum repository also contains MySQL 5.7. This course uses MySQL 8.0 as an example.
- Unless otherwise specified, the following slides are the inputs and outputs in MySQL 8.0.x.

Logging In to the MySQL Database and Initializing the Root User

- After the installation is complete, run the **mysql** command to log in to the database.

```
[root@localhost ~]# 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.
```

- After logging in to the database, run the following command to set a password for the **root** user. Otherwise, further operations cannot be performed.

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

- In MySQL 8.0.x, the default password of the **root** user is empty. The complete **mysql** command is **mysql -uroot -p**, where **root** is the user name and **-p** is followed by the password. An empty password sees no content after **-p**.
- MySQL databases have certain requirements on the password. It cannot be a weak password; otherwise, the password setting cannot take effect.

MySQL Configuration Files

- The main configuration file of MySQL is `/etc/my.cnf`. This file specifies the `/etc/my.cnf.d` directory for storing other configuration files.
- Basic parameters in the configuration file:
 - **basedir**: MySQL installation path.
 - **datadir**: directory for storing MySQL data.
 - **port**: listening port of the MySQL server.
 - **server_id**: server ID, which is used during cluster configuration.
 - **socket**: socket configuration. You can use this interface to quickly log in to MySQL instances of varied ports.
 - **tmdir**: directory for storing temporary files.

- The MySQL configuration file also contains some optimization parameters, such as **max_connections** (maximum number of connections), **max_connect_errors** (maximum number of incorrect connection attempts), and **wait_timeout** (number of seconds that the server waits for activity before closing a non-interactive connection).

MySQL Management Tool – mysqladmin

- mysqladmin is a MySQL command-line tool. It can run some basic commands for operations such as changing the password of the **root** user and viewing the MySQL status.

```
[root@localhost ~]# mysqladmin --help
.....
Where command is a one or more of: (Commands may be shortened)
create database[ame]  Create a new database
debug                Instruct server to write debug information to log
drop database[ename]  Delete a database and all its tables
extended-status       Gives an extended status message from the server
flush-hosts          Flush all cached hosts
flush-logs            Flush all logs
flush-status          Clear status variables
flush-tables          Flush all tables
flush-threads         Flush the thread cache
flush-privileges      Reload grant tables (same as reload)
kill id,id,...        Kill mysql threads
password [new-password] Change old password to new-password in current format
ping                 Check if mysql is alive
processlist           Show list of active threads in server
reload               Reload grant tables
refresh              Flush all tables and close and open logfiles
shutdown             Take server down
status               Gives a short status message from the server
.....
```

SQL Standards Compliance in MySQL

- In MySQL, SQL statements are case insensitive. You are advised to use uppercase letters for commands and lowercase letters for table names.
- SQL statements can be entered in a single line or multiple lines and end with a semicolon (;).
- In MySQL, use # or -- to comment out a single line, and use /*...*/ to comment out multiple lines.
- Each statement can be indented or wrapped as required. Spaces or Tab keys can be used for indentation.

SQL Classification

- DDL: Data Definition Language
 - CREATE, DROP, ALTER
- DML: Data Manipulation Language
 - INSERT, DELETE, UPDATE
- DQL: Data Query Language
 - SELECT
- DCL: Data Control Language
 - GRANT, REVOKE
- TCL: Transaction Control Language
 - COMMIT, ROLLBACK, SAVEPOINT

Basic MySQL Statements – Retrieve

- List the existing databases.
 - SHOW databases;
- View the current user.
 - SELECT user();
- View the current database.
 - SELECT database();

```
mysql> SHOW databases;
+-----+
| Database      |
+-----+
| information_schema |
| mysql          |
| performance_schema |
| sys            |
+-----+
4 rows in set (0.00 sec)
```

```
mysql> SELECT user();
+-----+
| user()      |
+-----+
| root@localhost |
+-----+
1 row in set (0.00 sec)
```

```
mysql> SELECT database();
+-----+
| database()  |
+-----+
| NULL        |
+-----+
1 row in set (0.00 sec)
```

Basic MySQL Statements – Retrieve

- List tables in the current database.
 - SHOW tables;
- View all data in a table.
 - SELECT * from *user*;
- View the value of an attribute in a table.
 - SELECT Host,User FROM *user*;

```
mysql> SHOW tables;
+-----+
| Tables_in_mysql |
+-----+
| columns_priv   |
| .....
```

```
mysql> SELECT Host,User FROM user;
+-----+-----+
| Host | User |
+-----+-----+
| localhost | mysql.infoschema |
| localhost | mysql.session |
| localhost | mysql.sys |
| localhost | root |
+-----+-----+
4 rows in set (0.00 sec)
```

Basic MySQL Statements – User Management

- Create a user.

```
CREATE USER [IF NOT EXISTS]
    user [auth_option] [, user [auth_option]] ...
    DEFAULT ROLE role [, role ] ...
    [REQUIRE {NONE | tls_option [[AND] tls_option] ...}]
    [WITH resource_option [resource_option] ...]
    [password_option | lock_option] ...
    [COMMENT 'comment_string' | ATTRIBUTE 'json_object']
```

- Modify users.

```
ALTER USER [IF EXISTS]
    user [auth_option] [, user [auth_option]] ...
    [REQUIRE {NONE | tls_option [[AND] tls_option] ...}]
    [WITH resource_option [resource_option] ...]
    [password_option | lock_option] ...
    [COMMENT 'comment_string' | ATTRIBUTE 'json_object']
```

```
ALTER USER [IF EXISTS]
    USER() user_func_auth_option
```

```
ALTER USER [IF EXISTS]
    user [registration_option]
```

```
ALTER USER [IF EXISTS]
    USER() [registration_option]
```

```
ALTER USER [IF EXISTS]
    user DEFAULT ROLE
    {NONE | ALL | role [, role ] ...}
```

Basic MySQL Statements – User Management

- Grant permissions to users.

```
GRANT  
    priv_type [(column_list)]  
    [, priv_type [(column_list)]] ...  
    ON [object_type] priv_level  
    TO user_or_role [, user_or_role] ...  
    [WITH GRANT OPTION]  
    [AS user  
        [WITH ROLE  
            DEFAULT  
            | NONE  
            | ALL  
            | ALL EXCEPT role [, role] ...  
            | role [, role] ...  
        ]  
    ]  
  
GRANT PROXY ON user_or_role  
    TO user_or_role [, user_or_role] ...  
    [WITH GRANT OPTION]  
  
GRANT role [, role] ...  
    TO user_or_role [, user_or_role] ...  
    [WITH ADMIN OPTION]
```

- Remove permissions from users.

```
REVOKE [IF EXISTS]  
    priv_type [(column_list)]  
    [, priv_type [(column_list)]] ...  
    ON [object_type] priv_level  
    FROM user_or_role [, user_or_role] ...  
    [IGNORE UNKNOWN USER]  
  
REVOKE [IF EXISTS] ALL [PRIVILEGES], GRANT OPTION  
    FROM user_or_role [, user_or_role] ...  
    [IGNORE UNKNOWN USER]  
  
REVOKE [IF EXISTS] PROXY ON user_or_role  
    FROM user_or_role [, user_or_role] ...  
    [IGNORE UNKNOWN USER]  
  
REVOKE [IF EXISTS] role [, role] ...  
    FROM user_or_role [, user_or_role] ...  
    [IGNORE UNKNOWN USER]
```

- Delete a user.

```
DROP USER [IF EXISTS] user [, user] ...
```

Basic MySQL Statements – Database Management

- In MySQL, tables are used to indicate relationships between data.
- Create a database.

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name  
[create_option] ...
```

- Modify a database.

```
ALTER {DATABASE | SCHEMA} [db_name]  
alter_option ...
```

- Delete a database.

```
DROP {DATABASE | SCHEMA} [IF EXISTS] db_name
```

- **DATABASE** and **SCHEMA** can replace each other.
- **IF NOT EXISTS** indicates a system check to confirm whether the data with a specified name already exists when creating a database. If this option is not used and a database with the same name already exists, the system reports an error.
- *create_option* includes CHARACTER SET (specifying a character set), COLLATE (collation rule), and ENCRYPTION (whether to encapsulate or not).

Basic MySQL Statements – Table Management

- A database is a repository for storing data.
- Create tables.
 - Manually create a table.

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name  
    (create_definition,...)  
    [table_options]  
    [partition_options]
```

- Copy or clone a table.

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name  
    { LIKE old_tbl_name | (LIKE old_tbl_name) }
```

- When creating a table, use *create_definition* to specify information about columns in the table, such as the column name, data type, and modifier.
 - Data types include numeric data, date and time, and string data. For details, see the official document at <https://dev.mysql.com/doc/refman/8.0/en/data-types.html>.
 - Modifiers are used to modify and supplement data, for example, whether data can be empty or primary keys.
- *table_options* can specify the engine type.
- Modifiers applicable to all types are NULL, NOT NULL, DEFAULT, PRIMARY KEY, UNIQUE KEY, and CHARACTER SET name.
- Modifiers applicable to numeric data are AUTO_INCREMENT and UNSIGNED.

Basic MySQL Statements – Table Management

- Modify a table.

```
ALTER TABLE tbl_name  
    [alter_option [, alter_option] ...]  
    [partition_options]
```

- Delete a table.

```
DROP [TEMPORARY] TABLE [IF EXISTS]  
tbl_name [, tbl_name] ...  
[RESTRICT | CASCADE]
```

- Query a table.

```
SHOW tables;
```

- Query the structure of a table.

```
DESC tbl_name;
```

- Check the status of a table.

```
SHOW TABLE STATUS LIKE 'tbl_name';
```

- When checking the table status, you can add \G to print the table in rows.

Basic MySQL Statements – Data Management

- The essence of data management is to modify the table content.
- Add (insert) data.

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name
[PARTITION (partition_name [, partition_name] ...)]
[(col_name [, col_name] ...)]
{ {VALUES | VALUE} (value_list [, (value_list)] ... }
[AS row_alias[(col_alias [, col_alias] ...)]]}
[ON DUPLICATE KEY UPDATE assignment_list]
```

- Modify data.

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
SET assignment_list
[WHERE where_condition]
[ORDER BY ...]
[LIMIT row_count]
```

- INSERT can insert one or more rows of data at a time.
- The syntax of INSERT can be simply denoted as `INSERT tbl_name [(col1,col2.....)] VALUES (val1,val2...),(val1,val2....).`
- If the data to be added to a row contains all columns, you do not need to specify the columns. Only when columns are overwritten do you need to specify the columns. The values must correspond to the columns.

Basic MySQL Statements – Data Management

- The essence of data management is to modify the table content.
- View data.

```
SELECT  
[ALL | DISTINCT | DISTINCTROW ]  
[HIGH_PRIORITY]  
[STRAIGHT_JOIN]  
[SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]  
[SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]  
select_expr [, select_expr] ...  
[into_option]  
[FROM table_references  
    [PARTITION partition_list]]  
[WHERE where_condition]  
[GROUP BY {col_name | expr | position}, ... [WITH ROLLUP]]  
[HAVING where_condition]  
[WINDOW window_name AS (window_spec)  
    [, window_name AS (window_spec)] ...]  
[ORDER BY {col_name | expr | position}  
    [ASC | DESC], ... [WITH ROLLUP]]  
[LIMIT {offset} row_count | row_count OFFSET offset]  
[into_option]  
[FOR {UPDATE | SHARE}  
    [OF tbl_name [, tbl_name] ...]  
    [NOWAIT | SKIP LOCKED]  
    | LOCK IN SHARE MODE]  
[into_option]
```

- Delete data.

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name  
[[AS] tbl_alias]  
[PARTITION (partition_name [, partition_name] ...) ]  
[WHERE where_condition]  
[ORDER BY ...]  
[LIMIT row_count]
```

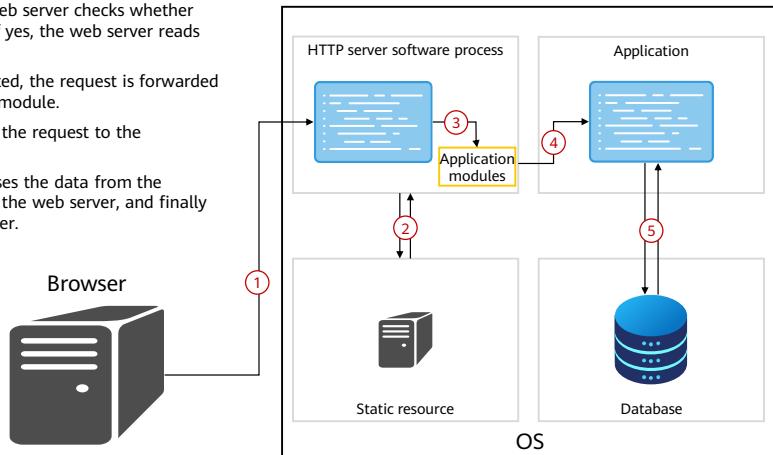
- SELECT has many options. The lab guide lists some commonly used queries.

Contents

1. Web Services
2. Apache Overview and Configurations
3. NGINX Overview and Configurations
4. DNS Overview and Configurations
5. MySQL Overview and Configurations
- 6. LAMP/LNMP**

Web Service Access Process

1. A user sends a request to the web server through a browser.
2. After receiving the request, the web server checks whether the static resource is requested. If yes, the web server reads and returns the static resource.
3. If the dynamic resource is requested, the request is forwarded to the corresponding application module.
4. The application module forwards the request to the corresponding application.
5. The application reads and processes the data from the database, returns the resource to the web server, and finally returns the resource to the browser.



Mainstream Implementations of Web Servers – LAMP and LNMP

- L stands for Linux, the OS on which web servers and application servers run.
- A and N stand for Apache and NGINX, respectively, which are used to implement web servers.
- M stands for MySQL, which is used to store data required by applications.
- P stands for PHP initially. Now it stands for various development languages starting with P, such as Python and Perl, which are used for application development.



106 Huawei Confidential



- M can also stand for MariaDB, which is an open source database developed by the founder of MySQL after MySQL is acquired by Oracle. The usage of MariaDB is basically the same as that of MySQL.
- In the following contents, P stands for PHP as an example.

Apache Dynamic Resource Configuration (PHP as an Example)

- Install PHP.
 - PHP applications require the support of the PHP program. Use the Yum repository to install PHP.
- Add PHP-related configurations to the mime module.
 - Add **AddType application/x-*httpd-php* .php** to the mime module to enable Apache to support PHP.
- Modify the configuration file.
 - Place the applications developed using PHP in the corresponding directory.

NGINX Dynamic Resource Configuration (PHP as an Example)

- Install PHP.
 - PHP applications require the support of the PHP program. You can use the yum repository to install PHP.
- Configure PHP permissions.
 - In the PHP configuration file **/etc/php-fpm.d/www.conf**, set the **user** and **group** parameters to the specified user in **nginx.conf**.
- Modify the configuration file.
 - Place the applications developed using PHP in the corresponding directory.

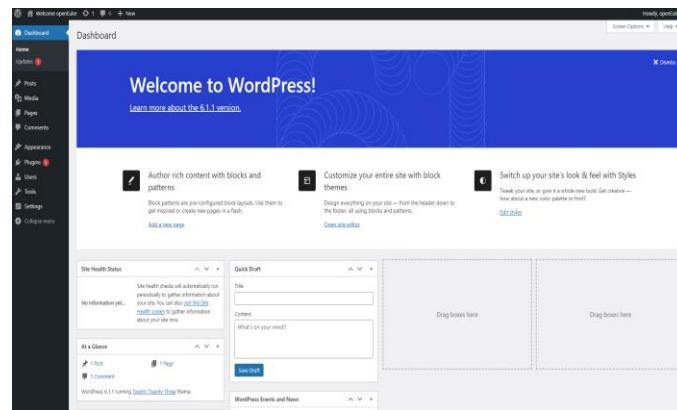
Connecting PHP to MySQL

- (Optional) Install the driver for connecting PHP to MySQL.
 - Install php-mysqlnd of the required version.
- MySQL grants access permissions to PHP hosts.
 - Modify the MySQL user permissions so that PHP hosts can access the MySQL database.
- Set the MySQL address, user name, and password in the PHP application file.

If you choose a different method for using PHP, you do not need to install the driver for PHP to connect to MySQL.

Typical LAMP Application – WordPress

- WordPress is the easiest and most popular way to create your own website or blog.
- WordPress is a content management system (CMS) written in PHP. It uses the MySQL database as the backend and can be accessed through HTTP or HTTPS.



Quiz

1. (Multiple-answer question) Which of the following are characteristics of MySQL databases?
 - A. MySQL is a commercial database.
 - B. MySQL databases are both relational and non-relational.
 - C. A large amount of contributed MySQL software is available.
 - D. MySQL works in client/server systems.

- CD

Summary

- This course covers two mainstream web service applications, Apache and NGINX, and details their working principles and basic configurations. Generally, a database server is deployed at the backend of a web server. This course uses MySQL as an example to describe how to perform add, delete, query, and modify operations using SQL statements. It also describes how to set up and configure the DNS server that is necessary for accessing the web server.

Acronyms and Abbreviations

| Acronym/Abbreviation | Full Spelling |
|----------------------|---------------------------|
| URL | uniform resource locator |
| URN | uniform resource naming |
| DSO | Dynamic Shared Object |
| MPM | Multi-Processing Module |
| DBS | Database System |
| SQL | Structured Query Language |

Thank you.

把数字世界带入每个人、每个家庭、

每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and organization for a fully connected, intelligent world.

Copyright©2024 Huawei Technologies Co., Ltd.
All Rights Reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.



openEuler Cluster Architecture



Foreword

- Single-host applications are insufficient for today's Internet demands, which has created space for cluster applications. This course describes how to configure tools for high availability clusters and load balancing clusters, including Linux Virtual Server (LVS), HAProxy, NGINX, and Keepalived.

Objectives

- Upon completion of this course, you will understand:
 - Basic configurations and deployments of LVS, NGINX, HAProxy and Keepalived
 - Basic operations of Redis

Contents

- 1. Cluster Overview**
2. Common Cluster Deployment Tools
3. Redis Overview

What Is a Cluster?

- A deployment of two or more computers used to complete a task. This popularity has grown to better handle the efficiency issues common with single-host deployments.
- Typical cluster types:
 - Load balancing (LB) cluster
 - Frontend access requests are forwarded to backend compute nodes based on various algorithms to balance the load of each node.
 - High availability (HA) cluster
 - HA clusters ensure that applications provide services without interruption. In the event of a fault, applications are automatically and quickly switched to another node.
 - High performance computing (HPC) cluster
 - HPC performs trillion operations per second using parallel algorithms and the parallel/distributed computing, which cannot be achieved by a single computer.

- A computer cluster is a group of computers that are loosely or closely connected but appear as a single system. Unlike grid computers, each node in a computer cluster performs same tasks that are scheduled by software.
- Components of a cluster are usually interconnected through a LAN, and each node (computer used as a server) runs its own OS instance. In most cases, all nodes in a cluster share the same hardware and OS.
- Clusters improve the performance, availability, and cost-effectiveness of computers.

Linux-based Load Balancing Cluster Software

- LVS
 - Open source software for load balancing at layer 4 (the transport layer)
- NGINX
 - Load balancing at layer 4 and layer 7 (the application layer)
- HAProxy
 - Load balancing at layer 4 and layer 7

- NGINX and HAProxy support layer-4 and layer-7 load balancing, though most commonly used for layer-7 load balancing.

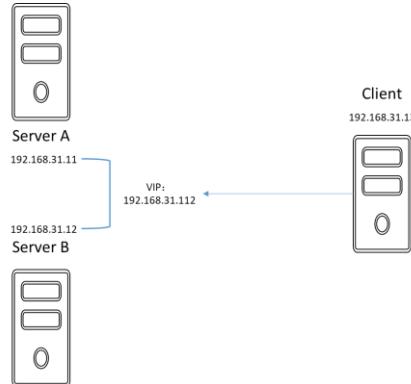
Linux-based HA Cluster Software

- Keepalived
 - A lightweight HA cluster deployment software in Linux. It is initially designed for LVS and then added with the Virtual Router Redundancy Protocol (VRRP) to implement functions such as cluster heartbeat detection and virtual IP address (VIP) failover.
- Heartbeat
 - Heartbeat does not manage VIP failover and outperforms Keepalived in managing resource services. Its configuration is more complex.
- Pacemaker
 - Pacemaker is an open source cluster resource manager in Linux applicable to clusters of any scale. It does not have the cluster heartbeat detection mechanism and depends on Corosync or Heartbeat to implement this function.

- In Linux, Keepalived and Heartbeat are commonly heartbeat mechanisms and both are supported by openEuler. Keepalived is easier to install, configure, use, and maintain than Heartbeat which is more complex but has more powerful functions and tools. Keepalived suits small- and medium-scale clusters, and Heartbeat for large-scale clusters.

Common Cluster Concepts – VIP and Address Failover

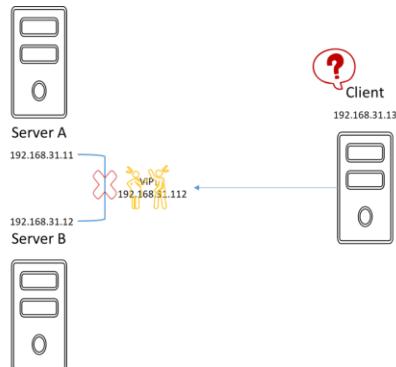
- VIP: virtual IP address. In an HA cluster, VIPs are used to provide services for external systems to avoid service access IP address changes when a host in the cluster is down.
- Address failover: A VIP is migrated from one host to another in an HA cluster.



- Case: Servers A and B form a cluster. The real IP addresses are 192.168.31.11 and 192.168.31.12 respectively. A VIP 192.168.31.112 is configured for them through HA cluster software. A client can ping 192.168.31.112 when server A or B is down.

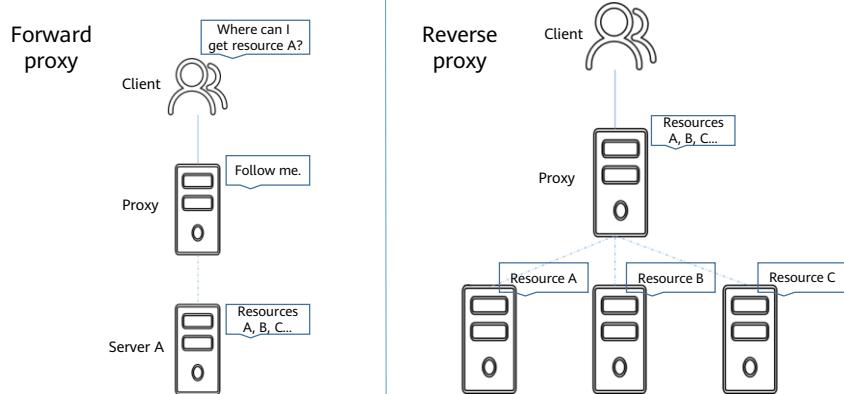
Common Cluster Concepts – Heartbeat and Split-brain

- Heartbeat: Hosts monitor the status of each other to determine which host provides services.
- Split-brain: Nodes preempt resources if the heartbeat communication between the nodes is interrupted or abnormal.



Common Cluster Concepts – Proxy

- Proxies can be classified into forward proxies and reverse proxies. Proxies are generally used in LB clusters.
- A forward proxy sits in front of one or more clients, and a reverse proxy sits in front of one or more servers.



Contents

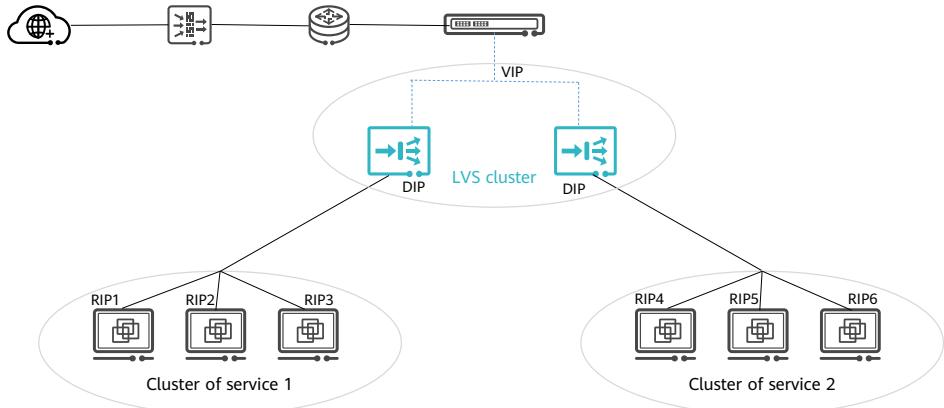
1. Cluster Overview

2. Common Cluster Deployment Tools

- LVS
 - NGINX
 - HAProxy
 - Keepalived

3. Redis Overview

Position of LVS in the Enterprise IT Architecture



12 Huawei Confidential

 HUAWEI

- VIP is a virtual IP address used by the LVS cluster to provide services to external systems.
- DIP is a director IP address for connecting to a service cluster. External networks cannot directly communicate with DIPs, and service requests are forwarded to DIPs through VIPs.
- RIP refers to real IP address of the service host and is reachable to DIPs.
- LVS can only implement LB scheduling and needs to work together with Keepalive to form an LVS cluster to eliminate single points of failure.

Components and Functions of LVS

- LVS consists of two parts:
 - IPVS: IP Virtual Server running in the kernel. It is implemented based on the kernel-space netfilter framework.
 - ipvsadm: LVS management tool running in user space. It can be used to configure LVS, such as adding, deleting, querying, and changing VIPs.
- ipvsadm command format:

```
ipvsadm -A|E virtual-service [-s scheduler] [-p [timeout]] [-M netmask] [--pe  
persistence_engine] [-b sched-flags]  
ipvsadm -D virtual-service  
ipvsadm -C  
ipvsadm -R  
ipvsadm -S [-n]  
ipvsadm -a|e virtual-service -r server-address [options]  
ipvsadm -d virtual-service -r server-address  
ipvsadm -L|l [virtual-service] [options]  
ipvsadm -Z [virtual-service]  
ipvsadm --set tcp tcpfin udp  
ipvsadm --start-daemon {master|backup} [daemon-options]  
ipvsadm --stop-daemon {master|backup}  
ipvsadm -h
```

Common Operations of ipvsadm

- Clears all rules.
 - ipvsadm -C
- Saves rules to a configuration file.
 - ipvsadm -Sn > /etc/sysconfig/ipvsadm
- Loads rules from a configuration file.
 - Ipvsadm -R < /etc/sysconfig/ipvsadm
- Lists existing rules.
 - Ipvsadm -Ln

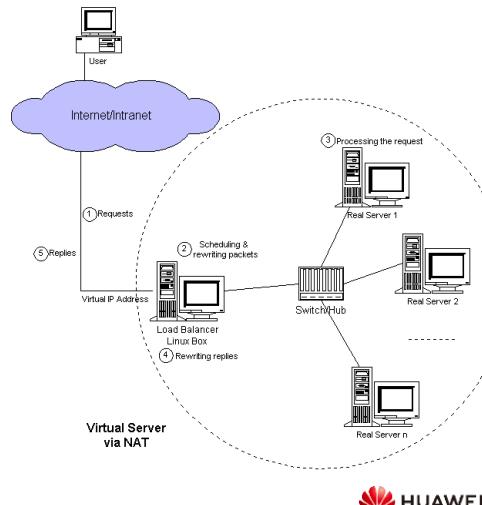
LVS Working Modes and Scheduling Algorithms

- LVS working modes:
 - LVS-NAT (network address translation)
 - LVS-DR (direct routing)
 - LVS-TUN (IP tunneling)
- LVS scheduling algorithms:
 - Round Robin (RR)
 - Weighted Round Robin (WRR)
 - Source Hashing (SH)
 - Destination Hashing (DH)
 - Least Connections (LC)
 - Weighted Least Connections (WLC)
 - Shortest Expected Delay (SED)
 - Never Queue (NQ)
 - Locality-Based Least Connections (LBLC)
 - Locality-Based Least Connections with Replication (LBLCR)
 - Weighted Failover (FO)
 - Overflow Connection (OVF)

- The NAT and DR modes are commonly used. Only the two modes are demonstrated in the lab guide.
- FO and OVF are new scheduling algorithms in version 4.15. They are seldom used and will not be described.

LVS Working Modes – NAT

- Assume that the user IP is 192.168.1.10, the LVS DIP is 10.0.0.20 (service port: 80), and the real server (RS) IP is 172.16.2.100 (service port: 8443).
 - The user sends a request to the LVS. The source address is 192.168.1.10 (user IP), the destination address is the LVS VIP, and the destination port is 80.
 - After receiving the request, the LVS schedules the request and rewrites the data packet. The source address remains unchanged, and the destination address is changed to 172.16.2.100 (RS IP) and the destination port is changed to 8443.
 - After receiving the request, the RS processes the request and changes the source address to 172.16.2.100 (RS IP), destination address to 192.168.1.10 (user IP), and source port to 8443 in the response packet.
 - After receiving the response packet, the LVS modifies the packet by changing the source address to its VIP, source port to 80, destination address to the user IP, and destination port to the port that sends the request.
 - The LVS sends the modified packet to the user.



- Image source: <http://www.linuxvirtualserver.org/VS-NAT.html>
- LVS-NAT forwards request packets after changing the destination addresses and ports based on configured rules and algorithms to the addresses and ports of different service hosts.
- In NAT mode, all packets pass through the LVS, making LVS a common bottleneck of the entire process. This mode is applicable to scenarios with a small number of backend servers.

LVS-NAT Configuration Example

- Enable the IP address forwarding function.

```
[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
```

- Add a cluster.

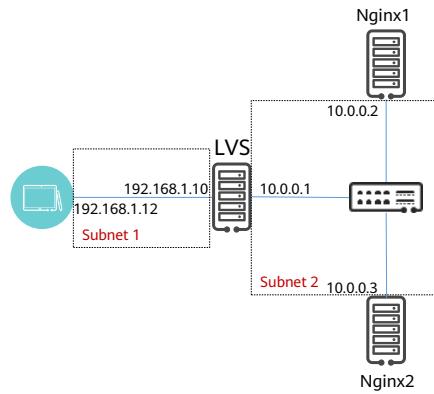
```
[root@Cluster1 ~]# ipvsadm -A -t 192.168.1.10:80 -s rr
```

- Add backend real servers (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 the status after configuration.

```
[root@Cluster1 ~]# ipvsadm -Ln  
IP Virtual Server version 1.2.1 (size=4096)  
Port 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
```

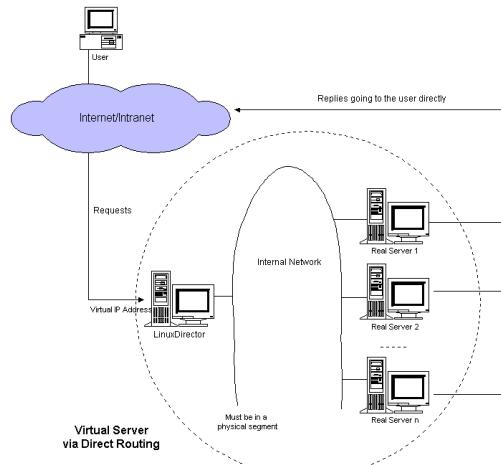


- In the **ipvsadm -A -t 192.168.1.10:80 -s rr** command, **-A** indicates that a cluster is added, **-t** indicates that TCP is used, and **-s rr** indicates that the scheduling algorithm is RR.

LVS Working Modes – DR

- In DR mode, all RSs need to be configured with two addresses: RIP and VIP.
- The LVS obtains the IP addresses and MAC addresses of all RSs through the Address Resolution Protocol (ARP). Therefore, the LVS and RSs must be on the same layer 2 network.
- After a user sends a request to the LVS, the LVS retains the source and destination IP addresses and ports, changes the source MAC address to its own MAC address and the destination MAC address to an RS MAC address, and forwards the request.
- When the RS returns a response packet, the source address is the VIP, the source port is the port that provides services, the source MAC address is its own MAC address, the destination address is the client IP, the destination port is the port that initiates the connection request, and the destination MAC address is the MAC address of the gateway.

18 Huawei Confidential



- Image source: <http://www.linuxvirtualserver.org/VS-DRouting.html>
- LVS-DR is the default mode. The essence is to re-encapsulate a request packet by changing the source MAC address to the MAC address of the DIP port and the destination MAC address to the MAC address of the RS RIP port. The source and destination IP addresses and ports remain unchanged. In this mode, only request packets pass through the LVS, and response packets do not. This reduces the LVS load and improves the performance of the entire process.
- The VIPs configured for the RSs must be the same as the LVS VIP. To prevent address conflicts, disable ARP advertisement and ARP reply functions of the NICs to which the VIPs of the RSs are assigned.
- Generally, all VIPs are configured on loopback interfaces.
- In DR mode, the LVS DIP and the RIPS of the RSs must be in the same network segment. The RIPS and VIPs can be in different network segments.
- Port numbers cannot be changed on the LVS, and the DIP and RIPS must be on the same network segment. In this case, large-scale or long-distance deployments are not recommended.

LVS-DR Configuration Example

- Configure VIPs on the LVS and RSs.

```
[root@Cluster ~]# nmcli connection add type dummy ifname dummy2  
ipv4.method manual ipv4.addresses 10.0.0.10/32
```

- Configure the ARP functions of the NICs to which the VIPs of the RSs are assigned.

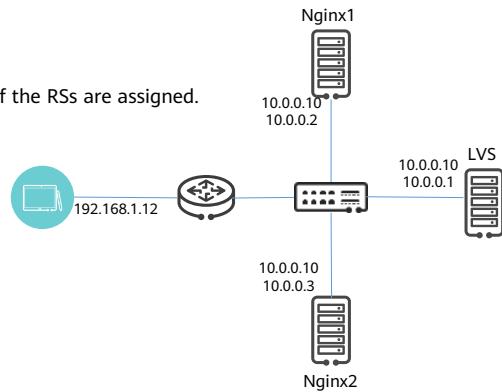
```
[root@Nginx1 ~]# echo 1 > /proc/sys/net/ipv4/conf/all/arp_ignore  
[root@Nginx1 ~]# echo 2 > /proc/sys/net/ipv4/conf/all/arp_announce  
[root@Nginx1 ~]# echo 1 > /proc/sys/net/ipv4/conf/dummy2/arp_ignore  
[root@Nginx1 ~]# echo 2 > /proc/sys/net/ipv4/conf/dummy2/arp_announce
```

- Configure forwarding rules on the LVS.

```
[root@Cluster ~]# ipvsadm -a -t 10.0.0.10:80 -r 10.0.0.2  
[root@Cluster ~]# ipvsadm -a -t 10.0.0.10:80 -r 10.0.0.3
```

- Add a cluster.

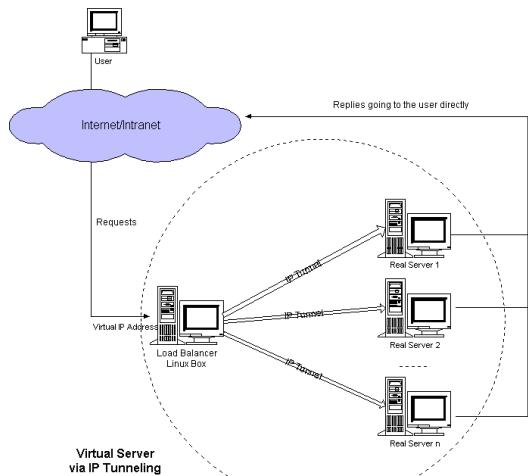
```
[root@Cluster1 ~]# ipvsadm -Ln  
.....  
-> RemoteAddress:Port      Forward Weight ActiveConn InActConn  
TCP 10.0.0.10:80 rr  
-> 10.0.0.2:80      Route 1    0      0  
-> 10.0.0.3:80      Route 1    0      0
```



- The ARP configuration commands modify only the temporary configuration file. After a device is restarted, the configuration becomes invalid. To make this configuration permanent, write the configuration into the configuration file.

LVS Working Modes – TUN

- The TUN mode is similar to the DR mode. The difference lies in that the DIP and RIPs can be in different network segments. After a request packet reaches the LVS, the LVS adds a tunnel header to the packet. The tunnel header contains the source address (DIP) and destination address (RIP).
- After the data packet passes through the tunnel and reaches the corresponding RS, the RS decapsulates the tunnel header to expose the packet header and packet content, and then processes and responds to the packet.
- The source address of the response packet is the VIP, and the destination address is the client IP.



- Image source: <http://www.linuxvirtualserver.org/VS-IPTunneling.html>
- The RSs use VIPs to respond. Therefore, VIPs need to be configured for the RSs.
- Advantage of the TUN mode: The LVS and RSs can be deployed in different network segments or WANs.

LVS Scheduling Algorithms – Static Algorithms

- There are four static scheduling algorithms:
 - Round Robin (RR)
 - The scheduler distributes the received requests to the RSs in the cluster in turn in sequence. Requests are evenly distributed, regardless of the number of server connections and server load.
 - Weighted Round Robin (WRR)
 - Different weights are assigned to RSs based on their processing capabilities. Servers with higher performance are allocated with more requests.
 - Source Hashing (SH)
 - The source IP address of a request is used as the hash key to identify a server in a statically assigned hash table. Requests from the same IP address are always forwarded to the same RS, implementing session binding.
 - Destination Hashing (DH)
 - The destination IP address of a request is used as the hash key to identify a server in a statically assigned hash table. Requests sent to the same destination address are always forwarded to the same RS, implementing load balancing in forward proxy caches.

21 Huawei Confidential



- The LVS has been integrated into the Linux kernel. You can run the **grep -i -C 10 ipvs /boot/config-5.10.0-60.18.0.50.oe2203.x86_64** command to view the algorithms supported by the LVS.

LVS Scheduling Algorithms – Dynamic Algorithms (1)

- There are eight dynamic scheduling algorithms:
 - Least Connections (LC)
 - Dynamically schedules network requests to the RS with the least connections. This algorithm can be used to balance loads in a cluster where servers have similar performance. It is suitable for applications using persistent connections.
 - Load calculation: Overhead = Number of active connections x 256 + Number of inactive connections
 - Weighted Least Connections (WLC)
 - The default LVS scheduling algorithm, WLC is recommended when the performance of servers in a cluster varies to ensure more active connections are scheduled to high-performance servers.
 - Load calculation: Overhead = (Number of active connections x 256 + Number of inactive connections)/Weight
 - Shortest Expected Delay (SED)
 - When WLC is used and the ratio of the number of connections to weight of each RS is the same, the next request may be allocated to any RS instead of the RS with the best performance. SED avoids this by ensuring the RS with the highest weight takes precedence.
 - Never Queue (NQ)
 - The requests in the first round are evenly allocated to all RSs, and SED is used later. This algorithm better utilizes idle servers.

- The LVS has been integrated into the Linux kernel. You can run the **grep -i -C 10 ipvs /boot/config-5.10.0-60.18.0.50.oe2203.x86_64** command to view the algorithms supported by the LVS.

LVS Scheduling Algorithms – Dynamic Algorithms (2)

- Locality-Based Least Connections (LBLC)
 - LBLC identifies the server that is recently used by the destination IP address in a request. If the server is available and not overloaded, this algorithm directs the request to the server. If the server is not identified or is overloaded and there are servers in half load, the request is forwarded to an available server based on the LC principle. LBLC is a destination IP-oriented load balancing algorithm and can be regarded as a dynamic version of DH. Forward proxy and web cache can be implemented by load.
- Locality-Based Least Connections with Replication (LBLCR)
 - LBLCR maintains mappings from a destination IP to a set of servers (LBLC maintains the mapping from a destination IP to a server). It identifies the server set mapped to the destination IP address in a request, selects a server that is not overloaded from the set based on the LC principle, and sends the request to the server. If all servers in this set are overloaded, it adds a server with least connections in the cluster to this server set and sends the request to the server. If the server set has not been modified for the specified time, the most loaded server is removed from the server set to avoid high degree of replication. This algorithm is also for destination IP load balancing. It solves the problem of load imbalance in LBLC and replicates requests from an RS with heavy load to an RS with light load. Forward proxy and web cache can be implemented by load.
- Weighted Failover (FO)
 - This algorithm traverses the RS linked list associated with the LVS and finds the RS that is not overloaded (**IP_VS_DEST_F_OVERLOAD** flag not set) and has the highest weight.

- The LVS has been integrated into the Linux kernel. You can run the **grep -i -C 10 ipvs /boot/config-5.10.0-60.18.0.50.oe2203.x86_64** command to view the algorithms supported by the LVS.

LVS Scheduling Algorithms – Dynamic Algorithms (3)

- Overflow Connection (OVF)
 - This algorithm implements overflow load balancing according to the number of active connections and weights of RSs. It traverses the RS linked list associated with the LVS and schedules new connections to the RS with the highest weight until the number of active connections exceeds its weight. In this case, this algorithm schedules new connections to the next RS with the highest weight. Available RSs must meet the following conditions:
 - Not overloaded (The **IP_VS_DEST_F_OVERLOAD** flag is not set.)
 - Number of active connections of the RS < RS weight
 - Weight ≠ 0

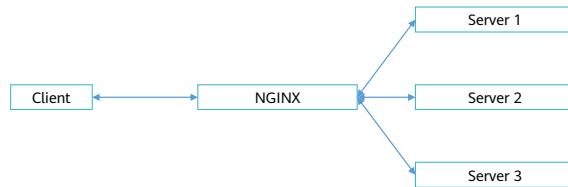
- The LVS has been integrated into the Linux kernel. You can run the **grep -i -C 10 ipvs /boot/config-5.10.0-60.18.0.50.oe2203.x86_64** command to view the algorithms supported by the LVS.

Contents

1. Cluster Overview
- 2. Common Cluster Deployment Tools**
 - LVS
 - NGINX
 - HAProxy
 - Keepalived
3. Redis Overview

NGINX Load Balancing

- NGINX provides layer-4 and layer-7 load balancing.
- NGINX provides multiple load balancing algorithms, such as round robin and weight.
 - Round robin: NGINX sends requests to different servers in turn so that each server provides services equally.
 - Weight: NGINX sends requests to servers based on the configured weights. Servers with good performance provide more services, and servers with poor performance provide fewer services.



Configuring Load Balancing with NGINX

- The load balancing function of NGINX is implemented by the upstream module.
- Configuration syntax:

```
upstream backend {  
    #balancing method  
    server backend1.example.com    weight=5;  
    server backend2.example.com:8080;  
    server unix:/tmp/backend3;  
    server backup1.example.com:8080  backup;  
    server backup2.example.com:8080  backup;  
}  
server {  
    location / {  
        proxy_pass http://backend;  
    }  
}
```

Algorithms Supported by NGINX

- RR
 - Requests are distributed evenly across the servers. This is the default algorithm of NGINX.
- LC
 - A request is sent to the server with the least number of active connections.
- IP Hash
 - The server to which a request is sent is determined from the source address.
- Generic Hash
 - The server to which a request is sent is determined from a user-defined key which can be a text string, variable, or a combination.
- Random
 - Each request will be passed to a randomly selected server. If the parameter **two** is added, NGINX randomly selects two servers and then chooses one of these servers using the specified method. If no method is specified, server weights are taken into account.

- NGINX also supports the Least Time algorithm, which is available only on NGINX Plus (commercial edition). It selects the server with the lowest average latency and the lowest number of active connections. The **header** or **last_byte** parameter needs to be included.

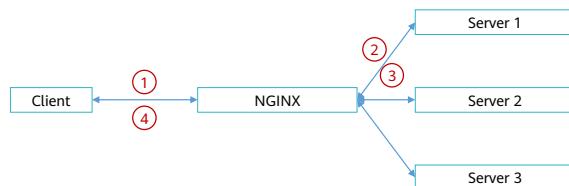
NGINX Load Balancing Command Parameters

- The upstream module provides other command parameters to configure load balancing.
 - **max_conns num**: Sets the maximum number of connections to a proxied server.
 - **down**: The server is down and NGINX does not forward requests to the server. This parameter is commonly used in IP Hash.
 - **backup**: The server is in the backup state. NGINX forwards requests to the server only when all other servers are faulty.
 - **maxfails** and **fail_timeout**: health check parameters. **maxfails** indicates the maximum failed attempts to pass a request to a backend server. The default value is 1, and if an attempt fails, NGINX does not forward requests to this server within the period specified by **fail_timeout**. After the specified period, NGINX attempts to forward new requests to the server. If this attempt still fails, NGINX does not forward requests to the server within this period again.
 - **slow_start**: health check parameter. If a server in the cluster is waiting for the period specified by **fail_timeout**, NGINX does not forward requests to the server immediately after it is recovered, but after the period specified by **slow_start**.

- **slow_start** is available in NGINX Plus (commercial edition).

NGINX Reverse Proxy

- Working process of the NGINX reverse proxy:
 1. A client sends a request to NGINX.
 2. After receiving the request from the client, NGINX forwards it to a backend server.
 3. The backend server returns the requested resource to NGINX.
 4. NGINX returns the resource to the client.
- In this process, the port used by the client to request resources from the proxy can be different from the port used by the server to provide services.



- NGINX can act as a proxy for protocols such as HTTP, HTTPS, TCP, WebSocket, gRPC, POP/IMAP, and RTMP.
- For example, the client uses port 80 to send requests to the proxy, but the server uses port 8080 to provide resources for the reverse proxy.

Configuring Reverse Proxies with NGINX

- The NGINX reverse proxy is configured in the **location** module.
- Use **proxy_pass** to set up a reverse proxy: **proxy_pass URL**
- If **proxy_pass** specifies a URL, the path in the **location** parameter is replaced.
- If **proxy_pass** does not specify a URL, the proxy retains the format of the client request and forwards the request to the server, or passes the full standardized request URI after modification.

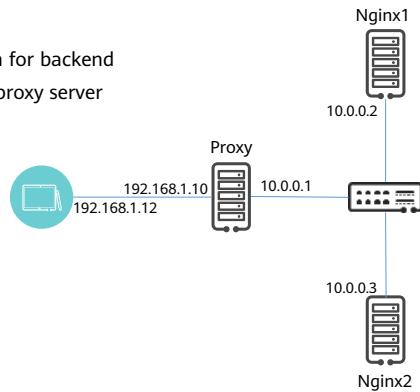
```
http {
    .....
    server {
        .....
        location /test {
            proxy_pass www.test.com:83/home;
        }
    }
}
```

- During reverse proxy configuration, the final forwarding path varies with the formats of the path in **location** and the URL in **proxy_pass** (slash usage). Assume that the address to be accessed is **http://192.168.1.123/test/index.html**:
 - **location /test/ { proxy_pass http://192.168.1.123/;}** The forwarding path is **http://192.168.1.123/index.html**.
 - **location /test/ { proxy_pass http://192.168.1.123;}** The forwarding path is **http://192.168.1.123/test/index.html**.
 - **location /test { proxy_pass http://192.168.1.123/;}** The forwarding path is **http://192.168.1.123//index.html**.
 - **location /test { proxy_pass http://192.168.1.123;}** The forwarding path is **http://192.168.1.123/test/index.html**.

NGINX Load Balancing and Reverse Proxy Configuration Example

- Requirements:
 - Use NGINX to set up a proxy server to provide the proxy function for backend servers Nginx1 and Nginx2. When a client accesses 10.0.0.1, the proxy server forwards the requests to the two Nginx servers in turn.
- Configurations on the proxy server:

```
upstream 10.0.0.1 {  
    server 10.0.0.2:80;  
    server 10.0.0.3:80;  
}  
server {  
    listen 80;  
    server_name localhost;  
    location / {  
        proxy_pass http://10.0.0.1;  
    }  
}
```



Contents

1. Cluster Overview

2. Common Cluster Deployment Tools

- LVS
- NGINX
- HAProxy
- Keepalived

3. Redis Overview

HAProxy Load Balancing

- One of the basic HAProxy functions which supports at least 10 scheduling algorithms
- Load balancing at layer 4 and layer 7
- Single-process and event-driven connection processing for better performance
- Access Control Lists (ACLs)
- GUI-based monitoring

Common HAProxy Scheduling Algorithms

- RR
 - Default scheduling algorithm.
- Source
 - Source address-based hash. It forwards requests with the same source address to the same server based on specified options.
- URI
 - Performs hashing based on the URI requested by the user and forwards the request to the specified server.
- HDR
 - Performs hashing based on the specified information in an HTTP header request and forwards the request to a selected backend server by weight. If there is no valid value, the default polling algorithm is used for scheduling.
- First
 - Depending on where the server is in the server list, this algorithm dispatches requests from top to bottom. If the number of connections to the first server reaches the maximum, new requests will be assigned to the next server.
- Leastconn
 - The dynamic LC algorithm preferentially forwards new requests to the server with the least active connections.

Features of HAProxy Scheduling Algorithms

- HAProxy scheduling algorithms allow manual weight adjustment, and such adjustment takes effect after HAProxy is restarted.
- RR, Leastconn, and hash algorithms support dynamic weight adjustment without restarting HAProxy.
- The algorithms capable of dynamic weight adjustment also support slow start.
- Hash algorithm objects can be client addresses, URL components, character elements in requests, header file values, POST parameters, and RDP cookies.
- Consistency hash is available.
- Multiple indicators can be used for making advanced load balancing policies.

HAProxy Configuration File

- The HAProxy configuration file consists of the following sections:
 - global
 - Defines global parameters. It is a process-wide configuration and is related to OS configurations.
 - default
 - Sets default parameters. These parameter settings apply to the **frontend**, **backend**, and **listen** sections.
 - frontend
 - Defines the frontend. You can configure ACLs in **frontend** to direct requests to matched **backend**.
 - backend
 - Defines the backend. One **backend** corresponds to one or more real servers.
 - listen
 - Combines the functionality of **frontend** and **backend**. It is usually used for status monitoring and backend server check.

HAProxy Configuration File Description (1)

```
global          # Sets global parameters.  
log 127.0.0.1 local2      # Syntax: log <address_1>[max_level_1] # Global log configuration. The keyword log is used to specify the  
local2 log device in the syslog service on 127.0.0.1 to record logs at the info level.  
chroot /var/lib/haproxy    # Change the working directory.  
pidfile /var/run/haproxy.pid # Current process ID file  
maxconn 4000            # Maximum number of connections. This setting is subject to the maxconn setting in the frontend section  
(preferred) or default section (less preferred).  
user haproxy           # User  
group haproxy          # User group  
daemon                # Runs HAProxy as a daemon process.  
stats socket /var/lib/haproxy/stats  
defaults  
mode http              # Default mode { tcp|http|health }. tcp: layer 4; http: layer 7; health: returns the word "OK".  
log global             # Applies the global log configuration.  
option httplog          # Enables logging of HTTP requests (not logged by default).  
option dontlognull      # Disables logging of null connections. In certain environments, an upstream load balancer or monitoring  
system will regularly connect to some component of this service to ensure that it is still alive. By default, even a simple port probe or scan will  
produce a log. If those connections pollute the logs too much, it is possible to enable option "dontlognull" to indicate that a connection on  
which no data has been transferred will not be logged, which typically corresponds to those probes. If the service does not have another  
upstream load balancer, this option is not recommended for environments such as web applications over the Internet where null connections  
could indicate malicious activities such as open port-scanning for vulnerabilities.
```

HAProxy Configuration File Description (2)

```
option http-server-close # Closes the HTTP channel after the request is complete.  
option forwardfor  
except 127.0.0.0/8 # If an application on the server wants to record the IP address of the client that initiates the request,  
configure this option on HAProxy. HAProxy will send the client IP address to the server and adds the "X-Forwarded-For" field to the  
HTTP request.  
option redispatch # In the case where a cookie is used, HAProxy inserts the server ID of the requested backend server into the  
cookie to prolong the session. Even if the backend server breaks down, the client cookie is not updated. If this option is used, the  
client request is forcibly directed to another backend server to ensure normal service running.  
retries 3 # Specifies the number of reconnection attempts to a backend server. If the number of retries exceeds this value, the  
backend server is marked as unavailable.  
timeout http-request 10s # HTTP request timeout interval  
timeout queue 1m # Timeout interval of a request in a queue  
timeout connect 10s # Connection timeout interval  
timeout client 1m # Client timeout interval  
timeout server 1m # Server timeout interval  
timeout http-keep-alive 10s # Sets the timeout interval for http-keep-alive.  
timeout check 10s # Check timeout interval  
maxconn 3000 # Maximum number of connections each process will accept.
```

HAProxy Configuration File Description (3)

```
frontend main *:80          # Uses port 80.
    acl url_static     path_end ~i .jpg .gif .png .css .js      # Defines an ACL.
    use_backend static  if url_static      # Forwards requests that match the url_static rule to
the backend named static.
    default_backend my_webserver      # Forwards requests to the backend named my_webserver by default.
backend static      # Enables static and dynamic resource separation. If url_path matches static files such as
.jpg, .gif, .png, .css, and .js, this backend is accessed.
    balance roundrobin      # Load balancing algorithm
    server static 127.0.0.1:80 check      # Deploys static files on the specified host. It can be the local host, another host, or the
Squid cache server.
backend my_webserver      # Defines a backend named my_webserver.
    balance roundrobin      # Load balancing algorithm
    server web01 172.31.2.33:80 check inter 2000 fall 3 weight 30      # Defines multiple backends.
    server web02 172.31.2.34:80 check inter 2000 fall 3 weight 30      # Defines multiple backends.
    server web03 172.31.2.35:80 check inter 2000 fall 3 weight 30      # Defines multiple backends.
```

HAProxy ACL Configuration

- HAProxy ACL syntax: acl <aclname> <criterion> [flags] [operator] [<value>]
 - <aclname>: user-defined rule name, which can contain letters (case sensitive), digits, and special characters.
 - <criterion>: matching criterion. HAProxy supports multiple matching criteria including the domain name, URL, source address, and destination address.
 - [flags]: criterion flag
 - [operator]: specific operator
 - [<value>]: object type

- Matching criteria include:
 - **hdr(host)**: full domain name. The content in the brackets can be another field in an HTTP request header, for example, User-Agent.
 - **hdr_reg(host)**: regular expression of the domain name
 - **hdr_dom(host)**: full domain name
 - **hdr_beg(host)**: domain name prefix
 - **hdr_end(host)**: domain name suffix
 - **hdr_len(host)**: domain name length
 - **hdr_sub(host)**: domain name substring
 - **hdr_dir**: domain name directory
 - **url_sub**: URL substring. Similar to hdr, there are eight matching methods.
 - **path_end**: path suffix. Similar to hdr, there are eight matching methods. **path** indicates the URL path of a request. The path starts from the first slash (/) and ends before the question mark (?) without the host part. In a complete URL
`<scheme>://<user>:<password>@<host>:<port>/<path>;<params>?<query>#<frag>`, **path** refers to the `/<path>;<params>` part. By default, the matched requests are scheduled to the same backend host.

- **base_beg**: base prefix. Similar to hdr, there are eight matching methods. **base** indicates the concatenation of the first host header and the path part of the request which starts from the first slash (/) and ends before the question mark (?). In a complete URL `<scheme>//<user>:<password>@<host>:<port>/<path>:<params>?<query>#<frag>`, **base** refers to the `<host>:<port>/<path>:<params>` part. This option is useful for virtual hosts.
 - **src**: source IP address
 - **src_port**: source port
 - **dst**: destination IP address
 - **dst_port**: destination port
 - **method**: HTTP request method (GET\POST...)
- Flag description:
 - **-i**: case insensitive
 - **-m**: uses a specific pattern matching method.
 - **-n**: forbids DNS resolution.
 - **-u**: forces the unique ID of an ACL. Otherwise, multiple ACLs with the same ID are matched.
 - **--**: forces end of flags. This flag is useful when a string looks like one of the flags.

- Operator Description
 - Integer:
 - **eq**: equal to; **ge**: greater than or equal to; **gt**: greater than; **le**: less than or equal to; **lt**: less than
 - Character string:
 - - **exact match (-m str)**: The extracted string must exactly match the patterns.
 - - **substring match (-m sub)**: The patterns are looked up inside the extracted string, and the ACL matches if any of them is found inside.
 - - **prefix match (-m beg)**: The patterns are compared with the beginning of the extracted string, and the ACL matches if any of them is found inside.
 - - **suffix match (-m end)**: The patterns are compared with the end of the extracted string, and the ACL matches if any of them is found inside.
 - - **subdir match (-m dir)**: The patterns are looked up anywhere inside the extracted string delimited with slashes (/). The ACL matches if any of them is found inside.
 - - **domain match (-m dom)**: The patterns are looked up anywhere inside the extracted string delimited with dots (.). The ACL matches if any of them is found inside.

- Value Description
 - **boolean**: false or true
 - **integer or integer range**: For example, 1024-32768 is used to match a port range; 192.168.0.1 or 192.168.0.1/24 is used to match an IP address or an IP address range.
 - **string**: full name, prefix, and suffix of a domain name, path, and URL
 - **regular expression**
 - **hex block**

GUI-based HAProxy Monitoring

| General process information | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|-----|-------|--------------|-----|-------|----------|-----|-------|-------|---------|-------|---------|-----|-----|--------|-----|------|----------|------------|-------|----------|--------------|------|-----|----------|-----|-----|----------|-------|--|--------|--|--|--------|--|--|----------|--|--|--------|--|--|-----|-----|-------|-----|-----|-------|-----|-----|-------|-------|-------|------|----|-----|-----|------|-----|------|-----|------|-------|--------|---------|------|-----|-----|-----|-----|----------|-------|----------|---|---|---|---|---|-------|----|----|-------|---------|---|---|---|----|---|---|---|---|---|------|----------|-------------|-----|---|---|---|---|----|---------|-------|---|---|---|---|-----|---|---|---|----|-------|---------|---|---|---|---|---|---|---|---|----------|----------|-------------|-----|---|---|---|----|----|---|-------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|----------|-------------|-----|---|---|---|---|----|---|-------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|----------|-------------|-----|---|---|---|---|----|---|-------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|----------|-------------|-----|---|---|---|---|----|---|-------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|----------|-------------|-----|---|---|---|---|----|---|---------|---|---|---|---|---|---|---|---|-----|---|---|---|---|---|---|---|---|---|---|---|----------|--|-----|---|---|--|---|----|---|
| <p>pid = 5772 process #1, nbproc = 1, nbthread = 4 uppers = 0d 0h0m39s system maxconn = unlimited, ulimit = 8041 maxsock = 8041; maxconn = 4000; maxpipes = 0 current conn = 1; current pipes = 0/0; conn rate = 1/sec; bit rate = 0.000 kbps Running tasks: 0/2; idle = 100 %</p> <p>Display option: Scope: [] • Hide DOWN servers • Disable refresh • Refresh now • CSV export • JSON export (schema)</p> <p>External resources: • Primary site • Updates (v2.4) • Online manual</p> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| main | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <table border="1"> <thead> <tr> <th colspan="3">Queue</th><th colspan="3">Session rate</th><th colspan="3">Sessions</th><th colspan="3">Bytes</th><th colspan="3">Denied</th><th colspan="3">Errors</th><th colspan="3">Warnings</th><th colspan="3">Server</th></tr> <tr> <th>Cur</th><th>Max</th><th>Limit</th><th>Cur</th><th>Max</th><th>Limit</th><th>Cur</th><th>Max</th><th>Limit</th><th>Total</th><th>LbTot</th><th>Last</th><th>In</th><th>Out</th><th>Req</th><th>Resp</th><th>New</th><th>Conn</th><th>Req</th><th>Retr</th><th>Redis</th><th>Status</th><th>LastChk</th><th>Wght</th><th>Act</th><th>Bck</th><th>Chk</th><th>Dwn</th><th>Downtime</th><th>Thrte</th></tr> </thead> <tbody> <tr> <td>Frontend</td><td></td><td></td><td></td><td></td><td></td><td>0</td><td>0</td><td></td><td>3 000</td><td>0</td><td></td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>OPEN</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> </tbody> </table> | | | | | | | | | | | | | | | | | | | Queue | | | Session rate | | | Sessions | | | Bytes | | | Denied | | | Errors | | | Warnings | | | Server | | | Cur | Max | Limit | Cur | Max | Limit | Cur | Max | Limit | Total | LbTot | Last | In | Out | Req | Resp | New | Conn | Req | Retr | Redis | Status | LastChk | Wght | Act | Bck | Chk | Dwn | Downtime | Thrte | Frontend | | | | | | 0 | 0 | | 3 000 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | OPEN | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Queue | | | Session rate | | | Sessions | | | Bytes | | | Denied | | | Errors | | | Warnings | | | Server | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Cur | Max | Limit | Cur | Max | Limit | Cur | Max | Limit | Total | LbTot | Last | In | Out | Req | Resp | New | Conn | Req | Retr | Redis | Status | LastChk | Wght | Act | Bck | Chk | Dwn | Downtime | Thrte | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Frontend | | | | | | 0 | 0 | | 3 000 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | OPEN | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <table border="1"> <thead> <tr> <th colspan="3">http_back</th><th colspan="3">Session rate</th><th colspan="3">Sessions</th><th colspan="3">Bytes</th><th colspan="3">Denied</th><th colspan="3">Errors</th><th colspan="3">Warnings</th><th colspan="3">Server</th></tr> <tr> <th>Cur</th><th>Max</th><th>Limit</th><th>Cur</th><th>Max</th><th>Limit</th><th>Cur</th><th>Max</th><th>Limit</th><th>Total</th><th>LbTot</th><th>Last</th><th>In</th><th>Out</th><th>Req</th><th>Resp</th><th>New</th><th>Conn</th><th>Req</th><th>Retr</th><th>Redis</th><th>Status</th><th>LastChk</th><th>Wght</th><th>Act</th><th>Bck</th><th>Chk</th><th>Dwn</th><th>Downtime</th><th>Thrte</th></tr> </thead> <tbody> <tr> <td>node1</td><td>0</td><td>0</td><td>-</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>-</td><td>0</td><td>0</td><td>?</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>8m39s UP</td><td>L4OK in 0ms</td><td>1/1</td><td>Y</td><td>-</td><td>0</td><td>0</td><td>0s</td><td>-</td></tr> <tr> <td>node2</td><td>0</td><td>0</td><td>-</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>-</td><td>0</td><td>0</td><td>?</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>8m39s UP</td><td>L4OK in 0ms</td><td>1/1</td><td>Y</td><td>-</td><td>0</td><td>0</td><td>0s</td><td>-</td></tr> <tr> <td>node3</td><td>0</td><td>0</td><td>-</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>-</td><td>0</td><td>0</td><td>?</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>8m39s UP</td><td>L4OK in 0ms</td><td>1/1</td><td>Y</td><td>-</td><td>0</td><td>0</td><td>0s</td><td>-</td></tr> <tr> <td>node4</td><td>0</td><td>0</td><td>-</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>-</td><td>0</td><td>0</td><td>?</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>8m39s UP</td><td>L4OK in 0ms</td><td>1/1</td><td>Y</td><td>-</td><td>0</td><td>0</td><td>0s</td><td>-</td></tr> <tr> <td>node5</td><td>0</td><td>0</td><td>-</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>-</td><td>0</td><td>0</td><td>?</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>8m39s UP</td><td>L4OK in 0ms</td><td>1/1</td><td>Y</td><td>-</td><td>0</td><td>0</td><td>0s</td><td>-</td></tr> <tr> <td>node6</td><td>0</td><td>0</td><td>-</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>-</td><td>0</td><td>0</td><td>?</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>8m39s UP</td><td>L4OK in 0ms</td><td>1/1</td><td>Y</td><td>-</td><td>0</td><td>0</td><td>0s</td><td>-</td></tr> <tr> <td>Backend</td><td>0</td><td>0</td><td>-</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>300</td><td>0</td><td>0</td><td>?</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>8m39s UP</td><td></td><td>6/6</td><td>6</td><td>0</td><td></td><td>0</td><td>0s</td><td>-</td></tr> </tbody> </table> | | | | | | | | | | | | | | | | | | | http_back | | | Session rate | | | Sessions | | | Bytes | | | Denied | | | Errors | | | Warnings | | | Server | | | Cur | Max | Limit | Cur | Max | Limit | Cur | Max | Limit | Total | LbTot | Last | In | Out | Req | Resp | New | Conn | Req | Retr | Redis | Status | LastChk | Wght | Act | Bck | Chk | Dwn | Downtime | Thrte | node1 | 0 | 0 | - | 0 | 0 | 0 | 0 | 0 | - | 0 | 0 | ? | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8m39s UP | L4OK in 0ms | 1/1 | Y | - | 0 | 0 | 0s | - | node2 | 0 | 0 | - | 0 | 0 | 0 | 0 | 0 | - | 0 | 0 | ? | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8m39s UP | L4OK in 0ms | 1/1 | Y | - | 0 | 0 | 0s | - | node3 | 0 | 0 | - | 0 | 0 | 0 | 0 | 0 | - | 0 | 0 | ? | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8m39s UP | L4OK in 0ms | 1/1 | Y | - | 0 | 0 | 0s | - | node4 | 0 | 0 | - | 0 | 0 | 0 | 0 | 0 | - | 0 | 0 | ? | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8m39s UP | L4OK in 0ms | 1/1 | Y | - | 0 | 0 | 0s | - | node5 | 0 | 0 | - | 0 | 0 | 0 | 0 | 0 | - | 0 | 0 | ? | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8m39s UP | L4OK in 0ms | 1/1 | Y | - | 0 | 0 | 0s | - | node6 | 0 | 0 | - | 0 | 0 | 0 | 0 | 0 | - | 0 | 0 | ? | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8m39s UP | L4OK in 0ms | 1/1 | Y | - | 0 | 0 | 0s | - | Backend | 0 | 0 | - | 0 | 0 | 0 | 0 | 0 | 300 | 0 | 0 | ? | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8m39s UP | | 6/6 | 6 | 0 | | 0 | 0s | - |
| http_back | | | Session rate | | | Sessions | | | Bytes | | | Denied | | | Errors | | | Warnings | | | Server | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Cur | Max | Limit | Cur | Max | Limit | Cur | Max | Limit | Total | LbTot | Last | In | Out | Req | Resp | New | Conn | Req | Retr | Redis | Status | LastChk | Wght | Act | Bck | Chk | Dwn | Downtime | Thrte | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| node1 | 0 | 0 | - | 0 | 0 | 0 | 0 | 0 | - | 0 | 0 | ? | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8m39s UP | L4OK in 0ms | 1/1 | Y | - | 0 | 0 | 0s | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| node2 | 0 | 0 | - | 0 | 0 | 0 | 0 | 0 | - | 0 | 0 | ? | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8m39s UP | L4OK in 0ms | 1/1 | Y | - | 0 | 0 | 0s | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| node3 | 0 | 0 | - | 0 | 0 | 0 | 0 | 0 | - | 0 | 0 | ? | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8m39s UP | L4OK in 0ms | 1/1 | Y | - | 0 | 0 | 0s | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| node4 | 0 | 0 | - | 0 | 0 | 0 | 0 | 0 | - | 0 | 0 | ? | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8m39s UP | L4OK in 0ms | 1/1 | Y | - | 0 | 0 | 0s | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| node5 | 0 | 0 | - | 0 | 0 | 0 | 0 | 0 | - | 0 | 0 | ? | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8m39s UP | L4OK in 0ms | 1/1 | Y | - | 0 | 0 | 0s | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| node6 | 0 | 0 | - | 0 | 0 | 0 | 0 | 0 | - | 0 | 0 | ? | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8m39s UP | L4OK in 0ms | 1/1 | Y | - | 0 | 0 | 0s | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Backend | 0 | 0 | - | 0 | 0 | 0 | 0 | 0 | 300 | 0 | 0 | ? | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8m39s UP | | 6/6 | 6 | 0 | | 0 | 0s | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <table border="1"> <thead> <tr> <th colspan="3">admin_stat</th><th colspan="3">Session rate</th><th colspan="3">Sessions</th><th colspan="3">Bytes</th><th colspan="3">Denied</th><th colspan="3">Errors</th><th colspan="3">Warnings</th><th colspan="3">Server</th></tr> <tr> <th>Cur</th><th>Max</th><th>Limit</th><th>Cur</th><th>Max</th><th>Limit</th><th>Cur</th><th>Max</th><th>Limit</th><th>Total</th><th>LbTot</th><th>Last</th><th>In</th><th>Out</th><th>Req</th><th>Resp</th><th>New</th><th>Conn</th><th>Req</th><th>Retr</th><th>Redis</th><th>Status</th><th>LastChk</th><th>Wght</th><th>Act</th><th>Bck</th><th>Chk</th><th>Dwn</th><th>Downtime</th><th>Thrte</th></tr> </thead> <tbody> <tr> <td>Frontend</td><td>1</td><td>2</td><td>-</td><td>1</td><td>2</td><td>3 000</td><td>31</td><td>31</td><td>7 975</td><td>391 286</td><td>0</td><td>0</td><td>0</td><td>15</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>OPEN</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr> <td>Backend</td><td>0</td><td>0</td><td>-</td><td>0</td><td>1</td><td>300</td><td>2</td><td>2</td><td>0</td><td>0s</td><td>7 975</td><td>391 286</td><td>0</td><td>0</td><td>0</td><td>2</td><td>0</td><td>0</td><td>0</td><td>0</td><td>8m39s UP</td><td></td><td>0/0</td><td>0</td><td>0</td><td></td><td>0</td><td>0s</td><td>-</td></tr> </tbody> </table> | | | | | | | | | | | | | | | | | | | admin_stat | | | Session rate | | | Sessions | | | Bytes | | | Denied | | | Errors | | | Warnings | | | Server | | | Cur | Max | Limit | Cur | Max | Limit | Cur | Max | Limit | Total | LbTot | Last | In | Out | Req | Resp | New | Conn | Req | Retr | Redis | Status | LastChk | Wght | Act | Bck | Chk | Dwn | Downtime | Thrte | Frontend | 1 | 2 | - | 1 | 2 | 3 000 | 31 | 31 | 7 975 | 391 286 | 0 | 0 | 0 | 15 | 0 | 0 | 0 | 0 | 0 | OPEN | | | | | | | | | Backend | 0 | 0 | - | 0 | 1 | 300 | 2 | 2 | 0 | 0s | 7 975 | 391 286 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 8m39s UP | | 0/0 | 0 | 0 | | 0 | 0s | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| admin_stat | | | Session rate | | | Sessions | | | Bytes | | | Denied | | | Errors | | | Warnings | | | Server | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Cur | Max | Limit | Cur | Max | Limit | Cur | Max | Limit | Total | LbTot | Last | In | Out | Req | Resp | New | Conn | Req | Retr | Redis | Status | LastChk | Wght | Act | Bck | Chk | Dwn | Downtime | Thrte | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Frontend | 1 | 2 | - | 1 | 2 | 3 000 | 31 | 31 | 7 975 | 391 286 | 0 | 0 | 0 | 15 | 0 | 0 | 0 | 0 | 0 | OPEN | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Backend | 0 | 0 | - | 0 | 1 | 300 | 2 | 2 | 0 | 0s | 7 975 | 391 286 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 8m39s UP | | 0/0 | 0 | 0 | | 0 | 0s | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

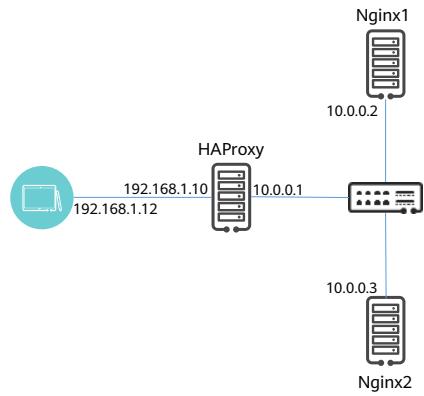
Invoking HAProxy ACLs

- Must comply with the following logical relationships:
 - AND: implicit, default one
 - OR: explicit with the "or" keyword or the "||" operator
 - Negation: with the exclamation mark ("!")
- ACL invoking statement formats:
 - `use_backend <backend> [{if | unless} <condition>]`: accepts a request if the condition is met.
 - `block {if | unless} <condition>`: blocks a request if the condition is met.
 - `http-request {allow | auth [realm <realm>] | redirect <rule> | deny [deny_status <status>]} [{if | unless} <condition>]`: used for layer-7 scheduling. **allow** indicates that the request is accepted. **auth** indicates user authentication information, and a prompt message is provided when **realm** is being specified. **redirect** indicates that the request is redirected. **deny** indicates that the HTTP request is rejected.
 - `tcp-request connection {accept | reject} [{if | unless} <condition>]`: used for layer-4 scheduling, indicating that a request is accepted or rejected if the condition is met.
 - `tcp-response connection < accept | reject | close> [{if | unless} <condition>]`: used for layer-4 scheduling, indicating that the response packet is accepted or rejected if the condition is met. **close** indicates that the connection to the server is closed immediately if the condition is met.

HAProxy Configuration Example

- Requirement: Use HAProxy to implement separated accesses to dynamic and static resources.
- HAProxy configurations:

```
global
.....
defaults
.....
frontend http
    bind 0.0.0.0:80
    mode http
    log global
    option httplog
    option httpclose
    acl php url_reg -i \.php$
    use_backend php-server if php
    default_backend static
backend php-server
.....
Backend static
.....
```



- Run the **haproxy -c -f /etc/haproxy/haproxy.cfg** command to check whether the configuration file is correct.

Comparison Between LVS, NGINX, and HAProxy

| | LVS | NGINX | HAProxy |
|--------------------------------|---------------|--|-------------------------------------|
| Network layer | Layer 4 | Layer 4 and layer 7 | Layer 4 and layer 7 |
| Efficiency | Very high | High | High |
| Tolerance to network stability | Low | High | High |
| Maintainability | Low | High | High |
| Fault detection and retry | Low | High | High |
| Session persistence | Supported | Not supported. This function is implemented through ip_hash. | Supported |
| Virtual host | Not supported | Supported | Supported |
| HTTPS transparent transmission | Supported | Supported | Supported in versions 1.5 and later |

Contents

1. Cluster Overview

2. Common Cluster Deployment Tools

- LVS
- NGINX
- HAProxy
- **Keepalived**

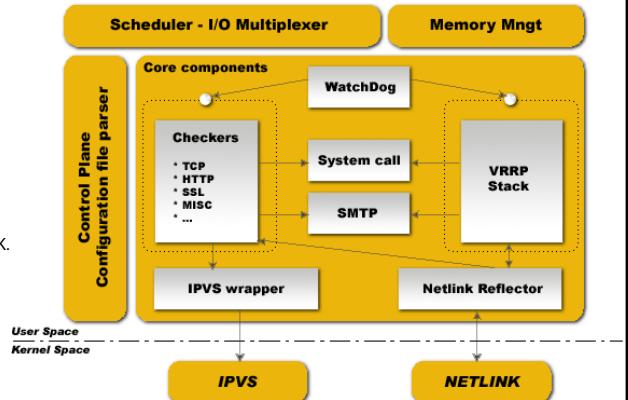
3. Redis Overview

Introduction to Keepalived

- Keepalived provides the following functions:
 - Health check on the LVS
 - Keepalived does not provide the load balancing function. It depends on the LVS running in the Linux kernel.
 - Failover processing in a load balancing cluster via VRRPv2
 - Keepalived has built-in health check tools to dynamically check the health of backend servers in a load balancing cluster, and works with VRRP to achieve high availability.

Keepalived Architecture

- Control panel
- Scheduler
- Memory management
- Core components
 - WatchDog: monitors processes.
 - Checkers: monitors and checks RSs.
 - VRRP: advertises VIPs.
 - Netlink Reflector: reflects network messages to NETLINK.
 - SMTP: email component that sends messages to the administrator.
 - IPVS wrapper: transfers IPVS rules.
 - System call: launches extra script during VRRP protocol state transition.



51 Huawei Confidential



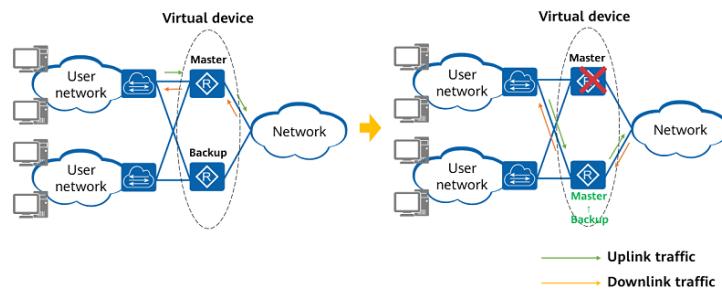
- Image source:
https://www.keepalived.org/doc/software_design.html#healthcheck-framework

Keepalived Health Checks

- Keepalived supports the following health check types:
 - TCP_CHECK
 - Implemented at layer 4. Keepalived sends a TCP connection request to the backend server. If the server does not reply to this request (timed-out), it is considered unavailable and removed from the server pool.
 - HTTP_GET
 - Implemented at layer 5. Keepalived sends an HTTP request to a specified URL and sums the result using MD5. If this sum does not match with the expected value, the server is removed from the server pool. In addition, an HTTP return code can be specified to determine whether the detection is successful. HTTP_GET can specify multiple URLs, which is useful in a scenario where a server has multiple virtual hosts.
 - SSL_GET
 - Same as HTTP_GET but uses an SSL connection to remote web servers.
 - MISC_CHECK
 - This check allows a user-defined script to be run as the health checker. The result 0 indicates the server status is normal. If **misc_dynamic** is set during configuration, the weight is automatically adjusted to the exit code minus 2.

Introduction to VRRP

- Virtual Router Redundancy Protocol (VRRP) deals with single points of failure (SPOFs) on static gateways and ensures uninterrupted operation of the entire network when some nodes break down.



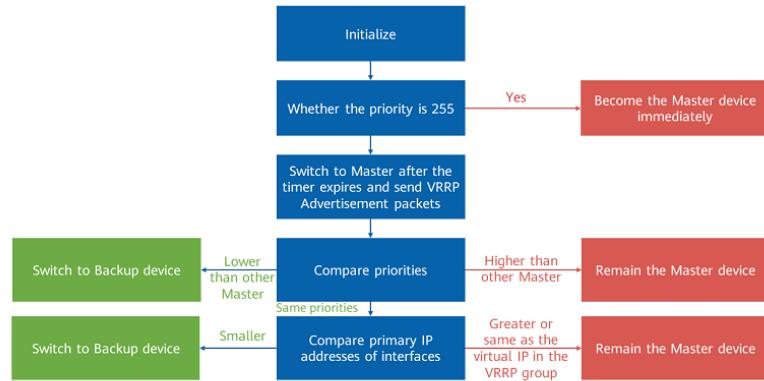
- Image source: <https://support.huawei.com/enterprise/en/doc/EDOC1100224965/>

VRRP Working Principles

- VRRP defines three states: Initialize, Master, and Backup. Only a device in the Master state can forward packets destined for a virtual IP address.

| State | Description |
|------------|--|
| Initialize | VRRP is unavailable. A device in the Initialize state does not process VRRP Advertisement packets. A device usually enters the Initialize state when it starts or detects a fault. |
| Master | A VRRP device in the Master state takes over all the forwarding tasks of the virtual routing device and sends VRRP Advertisement packets to the virtual router periodically. |
| Backup | A VRRP device in the Backup state does not take over the forwarding tasks of the virtual routing device, and receives the VRRP Advertisement packets from the master device periodically to determine whether the master device is working properly. |

VRRP Election Mechanism



55 Huawei Confidential



- Image source: <https://support.huawei.com/enterprise/en/doc/EDOC1100224965/>
- VRRP mechanism:
 - VRRP selects the master based on the priorities of devices in a VRRP group. The master device sends gratuitous ARP packets to notify connected devices or hosts of the virtual MAC address, and then forwards packets.
 - The master device periodically sends VRRP Advertisement packets to all backup devices in the VRRP group to advertise its configurations (such as the priority) and operating status.
 - If the master device fails, the backup device with the highest priority is elected as the new master.
 - After a master/backup switchover, the new master device immediately sends gratuitous ARP packets carrying the virtual MAC and virtual IP addresses to allow connected devices or hosts to update the corresponding MAC entries. After the update is complete, user traffic is switched to the new master device, which is transparent to users.
 - If the original master device recovers and it is the IP address owner (its priority is 255), it immediately switches to the Master state. If the original master device recovers and its priority is lower than 255, it switches to the Backup state, and its original priority is restored.
 - If the priority of a backup device is higher than that of the master device, VRRP determines whether to re-elect a new master, depending on the backup device's working mode (preemption or non-preemption).

Anti-brain-split Mechanism of Keepalived

- Third-party arbitration
 - Via an arbitration mechanism
 - Via arbitration software
- Multi-link heartbeat detection
 - From single heartbeat link to multiple heartbeat links
 - From single heartbeat type to multiple heartbeat types
- Custom detection script

Keepalived Configuration File Introduction – Global Configuration

- Keepalived has the following basic global configurations:

```
global_defs {  
    notification_email {  
        email ## Sets the administrator email for receiving notifications.  
        email ## Email  
    }  
    notification_email_from email ## Sets the email for sending notifications  
    smtp_server host ## IP address of the email server  
    smtp_connect_timeout num ## Timeout interval of the email server  
    router_id LVS_DEVEL ## ID of the physical node. Generally, the host name is used.  
    vrrp_skip_check_adv_addr  
    vrrp_strict  
    vrrp_garp_interval 0  
    vrrp_gna_interval 0  
}
```

- **vrrp_skip_check_adv_addr**: Checking all the addresses in a received VRRP advertisement can be time consuming. This flag skips check operations if the advertisement is from the same master router as the previous advertisement received.
- **vrrp_strict**: enforces strict VRRP compliance. Keepalived cannot be started in the following situations: 1. no VIP address. 2. unicast peers. 3. IPv6 addresses in VRRPv2.
- **vrrp_garp_interval 0**: interval between gratuitous ARP messages sent on an interface. The interval can be accurate to millisecond. (default: **0**)
- **vrrp_mcast_group4 224.0.0.18**: Multicast group to use for IPv4 VRRP advertisements. (default: **224.0.0.18**)
- **default_interface eth0**: default interface for static addresses. (default: **eth0**)

Keepalived Configuration File Introduction – VRRP Configuration

- Keepalived VRRP has the following basic configurations:

```
vrrp_instance VI_1 {
    state MASTER
    interface eth0
    packets.
    virtual_router_id 51
    priority 100
becomes the master router.
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 123456
    }
    virtual_ipaddress {
        192.168.200.16
    }
}
```

- **nopreempt**: stops the higher priority machine taking over the master role.
preempt is used by default.
- **mcast_src_ip**: source IP address of the heartbeat.

Keepalived Configuration File Introduction – LVS Configuration

- Keepalived LVS has the following basic configurations:

```
virtual_server 192.168.200.100 443 {
    delay_loop 6                      ## Health check interval
    lb_algo rr                          ## LVS scheduling algorithm
    lb_kind NAT                         ## LVS mode
    persistence_timeout 50              ## Persistence duration
    protocol TCP                        ## Protocol
    real_server 192.168.201.100 443 {
        weight 1                         ## Weight
        SSL_GET {
            url {
                path /
                digest ff20ad2481f97b1754ef3e12ecd3a9cc
            }
            connect_timeout 3             ## Timeout interval
            retry 3                      ## Number of retries
            delay_before_retry 3         ## Delay before a retry
        }
    }
}
```

Keepalived Health Check Configuration – TCP_CHECK

- TCP_CHECK configurations:

```
TCP_CHECK {  
    connect_ip <IP ADDRESS>; IP address to connect to. Default: IP address of the RS  
    connect_port <PORT>; port to connect to. Default: port on the RS to provide services  
    bindto <IP ADDRESS>; IP address of the interface that initiates the connection  
    bind_port <PORT>; source port that initiates the connection  
    connect_timeout <INT>; connection timeout interval. Default: 5 seconds  
    fwmark <INTEGER>; fwmark to mark all outgoing checker packets  
    warmup <INT>; random delay for maximum N seconds. It is useful to scatter multiple simultaneous  
checks to the same RS. 0: disabled  
    retry <INIT>; Retry count. Default: 1  
    delay_before_retry <INT>; Delay before a retry in seconds. Default: 1  
}
```

Keepalived Health Check Configuration – HTTP_CHECK and SSL_GET

- HTTP_CHECK or SSL_GET configurations:

```
HTTP_GET | SSL_GET {  
    url {  
        path <STRING>; URL to test  
        digest <STRING>; digest  
        status_code <INT>; status code  
    }  
    nb_get_retry <INT>; number of get attempts  
    delay_before_retry <INT>; delay before a retry  
    connect_ip <IP ADDRESS>; IP address to connect to. Default: IP address of the RS  
    connect_port <PORT>; port to connect to. Default: port on the RS to provide services  
    bindto <IP ADDRESS>; IP address of the interface that initiates the connection  
    bind_port <PORT>; source port that initiates the connection  
    connect_timeout <INT>; connection timeout interval. Default: 5 seconds  
    fwmark <INTEGER>; fwmark to mark all outgoing checker packets  
    warmup <INT>; random delay for maximum N seconds. It is useful to scatter multiple simultaneous checks to  
the same RS. 0: disabled  
}
```

- Digest calculation command: **genhash -s 172.17.100.1 -p 80 -u /index**

Keepalived Health Check Configuration – MISC_CHECK

- MISC_CHECK configurations:

```
MISC_CHECK {  
    misc_path <STRING>; Path to an external script or program  
    misc_timeout <INT>; Script execution timeout interval  
    user USERNAME [GROUPNAME]; user name or group name that the script should be run under. If GROUPNAME is not specified,  
    the group of the user is used.  
    misc_dynamic: dynamic weight adjustment based on the exit code from health checker  
    warmup <INT>; random delay for maximum N seconds. It is useful to scatter multiple simultaneous checks to the same RS. 0:  
    disabled  
}
```

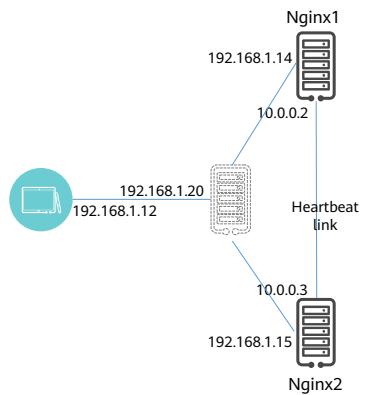
- Health check exit codes:
 - **0**: The health check is successful, and the weight remains unchanged.
 - **1**: The health check fails.
 - **2 - 255**: The health check is successful. The weight is set to the exit code minus 2. For example, if the exit code is 250, the weight is adjusted to 248.

NGINX HA Cluster Configuration with Keepalived

- Requirements: Nginx1 and Nginx2 form an HA cluster, detect each other's heartbeat through network segment 10, and provide services through floating IP address 192.168.1.10. If one host or the NGINX process on it is down, services must automatically switch over to the other host.
- Example configurations:

! Configuration File for keepalived

```
global_defs {  
    router_id Nginx1  
}  
  
vrrp_script nginx_check {  
    script "/home/check.sh"  
    interval 1  
    fail 3  
}  
  
vrrp_instance Nginx {  
    state MASTER  
    interface ens192  
    mcast_src_ip 10.0.0.2  
    virtual_router_id 51  
    priority 255  
    advert_int 1  
    authentication {  
        auth_type PASS  
        auth_pass Huawei@1  
    }  
    virtual_ipaddress {  
        192.168.1.20/24  
    }  
    track_script {  
        nginx_check  
    }  
}
```



NGINX Cluster Configuration with Keepalived + LVS

- Requirements: Nginx1 and Nginx2 form a cluster. Configure the LVS service at the frontend of the cluster to balance loads and use Keepalived to provide HA in the event an LVS container goes offline.
- Example configurations:

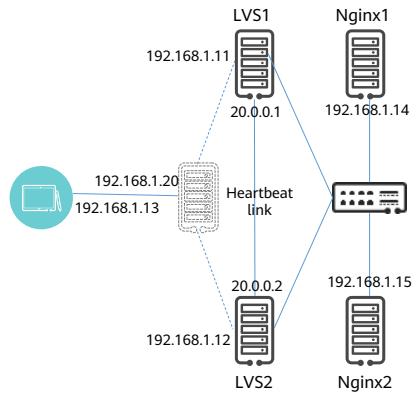
```
! Configuration File for keepalived

global_defs {
    router_id Cluster1
}

vrrp_instance Nginx {
    .....
    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
        }
    }
    .....
}
```



Contents

1. Cluster Overview
2. Common Cluster Deployment Tools
- 3. Redis Overview**

Introduction to Redis

- Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache, and message broker. It supports multiple types of data structures including strings, hash tables, lists, sets, sorted sets, bitmaps, HyperLogLogs, and geospatial indexes. Redis supports replication, LUA scripting, LRU eviction, transactions, and disk persistence at different levels, and provides high availability through Redis Sentinel and clustering.
- Redis' core capabilities:
 - In-memory database
 - Programmability
 - Scalability
 - Durability
 - Clustering
 - High availability

- This course focuses on the cache function.
- In-memory database: Redis is a popular data structure server supporting strings, hash tables, lists, sets, sorted sets, streams, etc.
- Programmability: server-side scripts in the language Lua and server-side storage using Redis functions
- Scalability: module APIs for building custom Redis extensions in C, C++, and Rust
- Durability: Datasets are retained in memory for quick access, and can be written to persistent storage for future use after a restart or system failure.
- Clustering: Hash-based sharding for horizontal scalability. Automatic sharding can be performed to scale up a cluster to up to millions of nodes.
- High availability: replication with automatic failover, applicable to both standalone deployment and cluster deployment

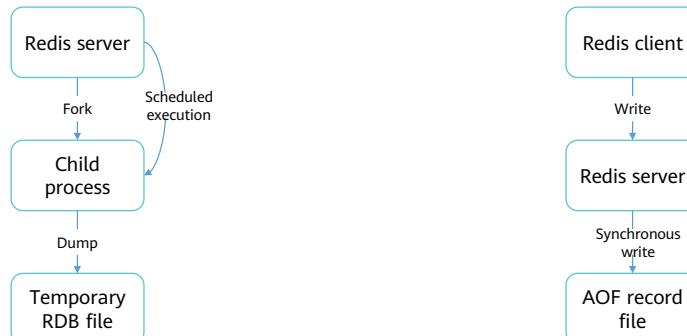
Redis Data Types

- Strings: most basic type in Redis. One key corresponds to one value.
- Lists: lists of strings, which are sorted in the insertion sequence.
- Sets: unsorted sets of strings. Members in a set must be unique.
- Hash tables: mapping tables where fields and values are strings. This type is applicable to object storage.
- Sorted sets: Similar to sets, sorted sets are collections of non-repeating strings sorted by Redis in ascending order.
- Streams: message queues.
- Geospatial indexes: geographical location information, such as coordinates and distances between locations.
- Bitmaps: binary bit arrays represented by 0s and 1s.
- Bitfields: bit fields consisting of one or more adjacent bits.
- Hyperlog: cardinality statistics

- In this course, you only need to get a basic understanding of the Redis data types.

Redis Data Persistence Methods

- Redis Database (RDB): Writes dataset snapshots in the memory to disks at a specified interval.
- Append Only File (AOF): Logs write and delete operations processed by the server.



- RDB method: Call the SAVE or BGSAVE commands.
- AOF method: Modify parameters in the configuration file.
 - appendfsync always # Synchronizes data into the AOF every time data is modified.
 - appendfsync everysec # Synchronizes data every second. (Default)
 - appendfsync no # Never synchronization. This is efficient but data is not persisted.
- RDB advantages: RDB is a compact single-file point-in-time snapshot of all Redis data that allows faster data restoration than AOF, making it ideal for data backup, full replication, and disaster recovery.
- RDB disadvantages: Each time a BGSAVE command is called, RDB needs to fork() using a child process. This is time consuming, and real-time or second-grade persistence is not supported. In addition, due to continuous Redis iteration, RDB formats of an earlier version may not be compatible with RDB files of a later version.
- AOF advantages: higher data consistency and integrity and second-grade data persistence.
- AOF disadvantages: AOF files are usually bigger than the equivalent RDB files for the same data set. Data recovery is also slower.

Common Redis Operations

- **set key value**: adds a key value.
- **exists key**: checks whether the key exists.
- **get key**: gets the value corresponding to the key.
- **keys ***: lists all keys in the current database.
- **del key**: deletes a key.
- **expire key**: sets the expiration time of the key.
- **ttl key**: queries the time to live of the key.
- **move key 0-15**: moves a key from a database to the specified database. *0-15* is the database ID range.
- **select 0-15**: selects the Redis logical database having the specified zero-based numeric index.
- **dbsize**: checks the number of keys in the current database.
- **flushdb**: deletes all keys from the current database.
- **flushall**: deletes keys from all databases.

- **unlink key**: deletes the key in non-blocking mode. This command is recommended when the key value is large.

Querying Redis Commands (1)

- Query from the official manual

- <https://redis.io/commands/>

The screenshot shows a search interface for Redis commands. At the top, there is a search bar labeled 'Command' containing 'SET'. To the right of the search bar is a dropdown menu labeled 'Data type' with 'String' selected. Below the search bar, there are two rows of four command cards each. The first row contains: 'GETSET' (Set the string value of a key and return its old value), 'MSET' (Set multiple keys to multiple values), 'MSETNX' (Set multiple keys to multiple values, only if none of the keys exist), and 'PSETEX' (Set the value and expiration in milliseconds of a key). The second row contains: 'SET' (Set the string value of a key), 'SETEX' (Set the value and expiration of a key), 'SETNX' (Set the value of a key, only if the key does not exist), and 'SETRANGE' (Overwrite part of a string at key starting at the specified offset).

- Image source: the official manual
- Command syntax description: The command name are uppercase letters, and options and parameters are lowercase letters. Parameters in [] are optional, and others are mandatory. Parameters separated by vertical bars (|) indicate available parameters.
- In Redis, commands are case insensitive, but key names are case sensitive.

Querying Redis Commands (2)

- Query on the CLI
 - Enter **HELP @ *data_type*** in the Redis CLI to query all commands supported by the specified data type.

```
127.0.0.1:6379> HELP @string
APPEND key value
summary: Append a value to a key
since: 2.0.0

BITCOUNT key [start] [end]
summary: Count set bits in a string
since: 2.6.0

BITFIELD key [GET type offset] [SET type offset value] [INCRBY type offset increment] [OVERFLOW WRAP|SAT|FAIL]
summary: Perform arbitrary bitfield integer operations on strings
since: 3.2.0

BITOP operation destkey key [key ...]
summary: Perform bitwise operations between strings
since: 2.6.0

BITPOS key bit [start] [end]
summary: Find first bit set or clear in a string
since: 2.6.7

DECR key
summary: Decrement the integer value of a key by one
since: 1.0.0

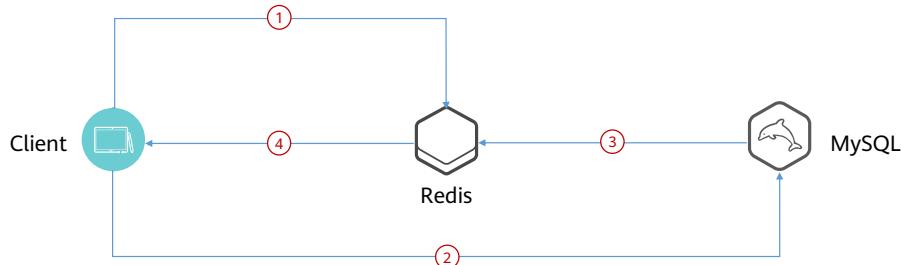
DECRBY key decrement
summary: Decrement the integer value of a key by the given number
since: 1.0.0

GET key
summary: Get the value of a key
since: 1.0.0
```

- Image source: the official manual

Working Principle of Redis as a Cache

- 1. When reading data, the client preferentially queries data in Redis. If the data is hit, Redis returns the result.
- 2. If not hit, the client queries MySQL, and MySQL returns the result.
- 3. When MySQL returns the result, the data queried by the client is written to Redis.
- 4. If the same data is read again, the data is directly returned from Redis.



72 Huawei Confidential



- Step 3 can be implemented using a third-party tool or server code, which is not discussed in this course.

Redis Use Case – Redis as the Cache of WordPress

Redis Object Cache

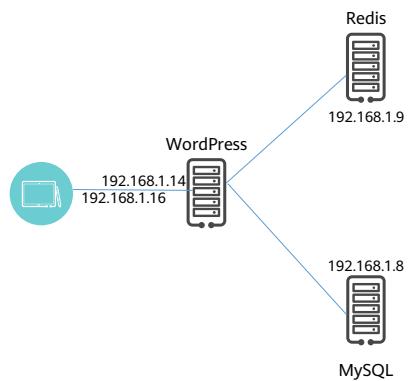
Overview Metrics Diagnostics

Overview

Status: ✓ Connected
Drop-in: ✓ Valid
Filesystem: ✓ Writeable

Connection

Client: Predis (v2.1.1)
Host: 192.168.1.9
Port: 6379
Username: H
Password: *****
Database: 0
Connection Timeout: 1s
Read Timeout: 1s
Redis Version: 6.2.7



Quiz

1. (True or False) Redis and MySQL are of the same type and can replace each other.
 - A. True
 - B. False
2. (Single Choice) A user uses the LVS to implement service load balancing. To hide the exposed ports of the service host, which of the following modes is recommended?
 - A. LVS-NAT
 - B. LVS-TUN
 - C. LVS-DR
 - D. LVS-TUN+

- B
- A

Summary

- This chapter describes the installation and configuration of mainstream cluster software, scheduling algorithms supported by the cluster software, and the benefit of each algorithm.

Acronyms and Abbreviations

| Acronym/Abbreviation | Full Spelling |
|----------------------|---|
| LB | load balancing |
| HA | high availability |
| HPC | high-performance computing |
| VIP | virtual IP |
| DIP | director IP |
| RIP | real IP |
| RR | round robin |
| WRR | weighted round robin |
| LC | least connections |
| WLC | weighted least connections |
| LBLC | locality-based least connections |
| LBLCR | locality-based least connections with replication |
| DH | destination hashing |
| SH | source hashing |

Thank you.

把数字世界带入每个人、每个家庭、

每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and organization for a fully connected, intelligent world.

Copyright©2024 Huawei Technologies Co., Ltd.
All Rights Reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.



openEuler Storage Management



Foreword

- Storage is an important concept in a computer system, responsible for storing and managing data. As information technology advances and enterprise storage demands evolve, storage technologies and architectures continue to innovate. In this course, we will explore the basic concepts and practical applications of various storage types.

Objectives

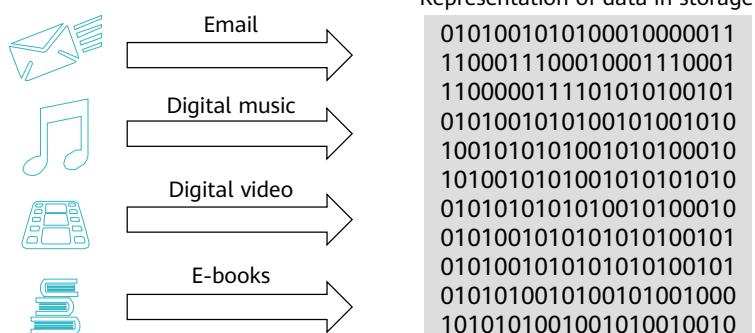
- On completion of this course, you will understand:
 - Basic concepts of different storage types.
 - NAS base principles and usage.
 - SAN types and application scenarios.
 - Basic principles and common volume modes of GlusterFS.

Contents

- 1. Basic Storage Concepts**
2. Storage Types
3. Introduction to Shared Storage
4. Introduction to Distributed Storage

Definition of Data

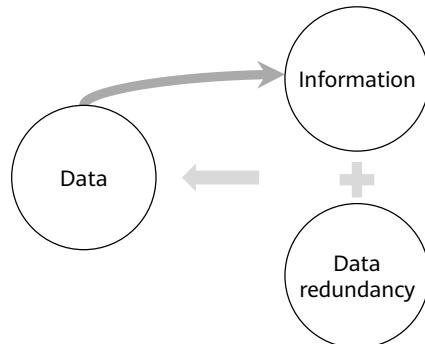
- The Storage Networking Industry Association (SNIA) defines data as the numerical representation of everything.



- Data is symbolic records that describe objective matters. It is unprocessed, raw material. A group of numbers, a piece of text, or an image can be seen as data.
- Computers use binary, where everything is represented as 0s and 1s. This means any data to be stored and accessed by a computer is represented by a sequence of these numbers.

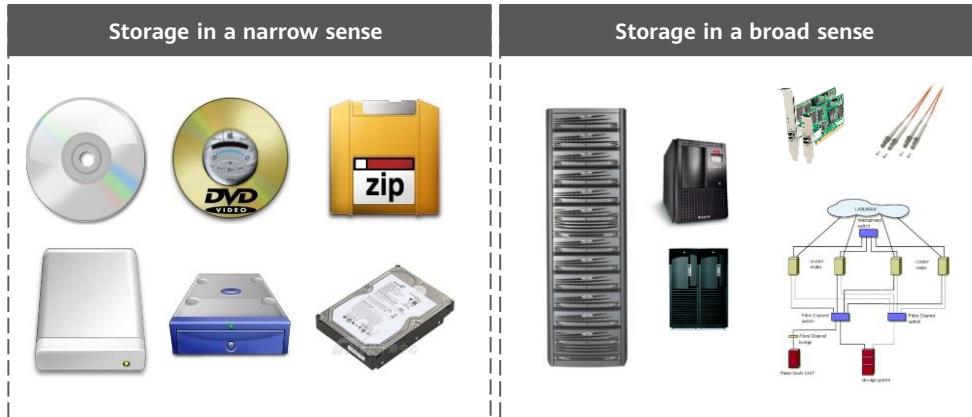
Definition of Information

- Information is the interpretation of data. In essence, information is processed, logically relational data that provides context, relevance, and purpose.



Introduction to Storage

- Data storage, or storage for short, refers to the process of saving data files or information on electronic devices.



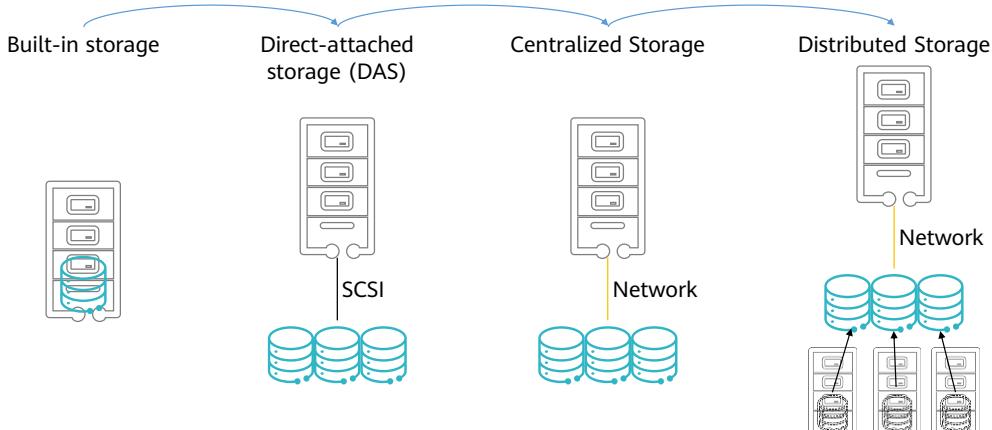
7 Huawei Confidential

HUAWEI

- "Storage" refers to the media that save user data or information in computer systems. Storage can refer to hardware devices such as hard disk drives (HDDs), solid state drives (SSDs), or USB flash drives, or software systems such as cloud storage or databases.
- "Store" refers to the process of saving data or information on physical devices, such as HDDs and flash memory, ensuring that data will not be lost even when the computer is shut down or powered off.
- Internal storage generally refers to storage devices inside a computer, for example, HDDs, SSDs, and memory. External storage refers to external devices that store data, for example, USB flash drives, external hard drives, and optical discs.
- Hard drives are the basic medium for storage. Early storage methods include paper and tape data storage.
- Storage media include:
 - HDDs
 - SSDs
 - Optical discs
 - Tapes

- Clouds

Evolution of Storage Architectures



8 Huawei Confidential



- While punched paper tapes were the earliest storage devices, the widespread adoption of internal hard drives within computers and servers truly marked the beginning of the storage architecture evolution.
- Standalone storage era: Hard drives were the built-in storage within servers. The physical space of servers limited drive capacity and hindered performance improvements. This led to the emergence of the DAS technology. In this era, enterprises had smaller amounts of data and simple storage requirements. External storage devices were directly attached to internal server buses to address the demand for storage expansion and high-performance data transfer on individual servers.
- Centralized storage era: The growing scale of enterprise data posed higher storage requirements, but this era was defined by low utilization and independent nature of servers and storage devices. Cross-OS file and application sharing, along with improved storage scalability, became pressing concerns. The standalone storage space offered by DAS could no longer support the service requirements, leading to the emergence of network storage device solutions based on LANs.
- Distributed storage era: The rise of the Internet and cloud computing saw an explosion of data that has become increasingly complex and diverse. This drove a surge in demand for elastic scalability and heterogeneous storage. Distributed management software enables integrated storage space across multiple storage

nodes, presenting a unified storage space to frontend application servers to cost-efficiently scale systems.

- Storage architectures can be classified by their access method:
 - Direct storage, like hard drives and optical discs, where storage media are fixed and can be accessed only through physical connection.
 - Network storage, like network drives and network-attached storage (NAS), allows data to be accessed and transferred through network storage protocols.
 - Distributed storage, like distributed file systems and distributed databases, where data distributed across multiple locations is managed and accessed on a unified platform.

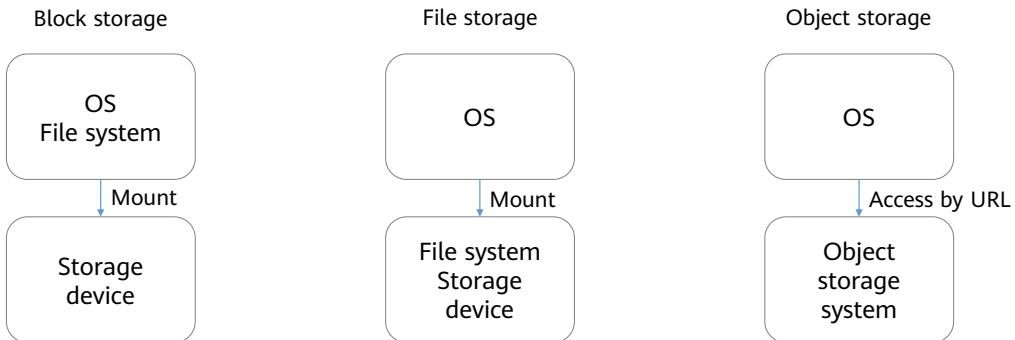
Contents

1. Basic Storage Concepts
- 2. Storage Types**
3. Introduction to Shared Storage
4. Introduction to Distributed Storage

Storage Types by Usage

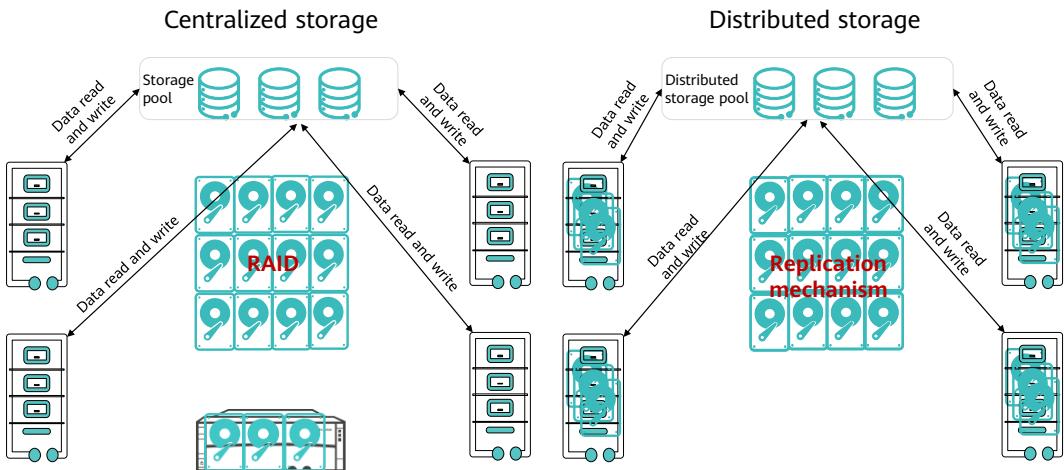
- Storage is classified based on how it is used:
 - Shared storage means a storage device is used by multiple servers. Technologies typically used for shared storage include:
 - Storage area network (SAN)
 - NAS
 - Distributed storage
 - Exclusive storage refers to storage devices used only by one server. Typical options include:
 - Built-in hard drives
 - DAS

Storage Types by Storage Method



- Block storage
 - The raw drive space is mapped to the host. The mounted raw drive is partitioned and formatted by the OS before it can be used.
- File storage
 - File storage is usually represented by directories and files (for example, **C:\Users\Downloads\text.doc**). Data is stored and accessed as files and organized in directories.
- Object storage
 - Object storage is usually represented by URLs (for example, **https://www.abc.com/app?v=nAKxJbcec8U**). Data and its metadata are packed as objects and stored in a massive storage pool.

Storage Types by Storage Media Location



12 Huawei Confidential



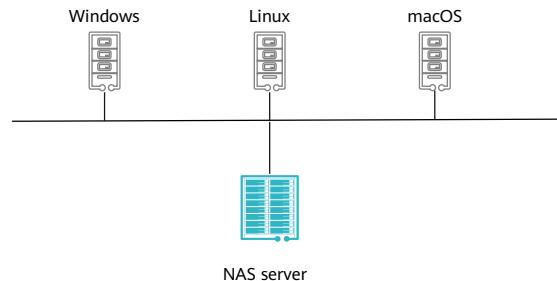
- In centralized storage, data is stored on a central node (storage pool) consisting of one or more servers. The central node also runs all services, manages data from branch nodes, allocates data on demand, and controls data access through a single controller.
- DAS, NAS, or SAN can be used selectively to implement centralized storage as required.
- Distributed storage combines the drive space of multiple servers within an enterprise network, and creates a virtual storage device that stores data across various locations.

Contents

1. Basic Storage Concepts
2. Storage Types
- 3. Introduction to Shared Storage**
4. Introduction to Distributed Storage

Introduction to NAS

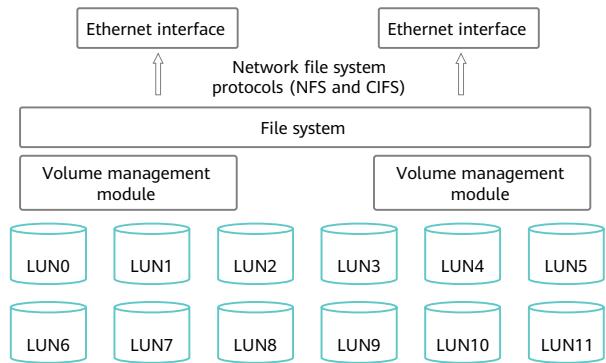
- NAS can be used in:
 - Shared storage
 - File storage



- NAS is a network storage technology that offers a simple, secure, and scalable method for file sharing and data storage.
- A NAS network includes hard drives that are connected to one or more servers or client devices to provide storage space.
- It is typically built around a core storage device, which communicates with other devices within the network. The core is usually controlled by a dedicated NAS server or embedded system. It is capable of automated management and backup of data and provides an intuitive user interface for data access and sharing.
- A NAS network also integrates components like storage protocols (SMB, NFS, iSCSI, etc.) and network interfaces to enable connection and data exchange. In addition, management tools and security measures are used to ensure data security and integrity.

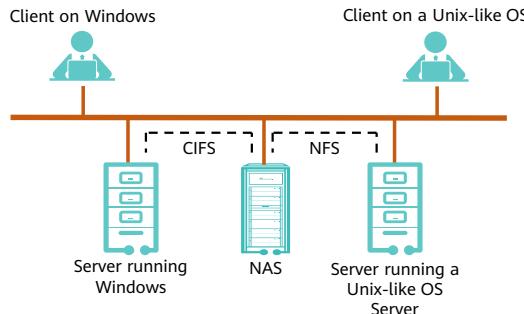
NAS Device Architecture

- A NAS device comprises:
 - NAS engine (CPU and memory)
 - One or more NICs
 - OS for NAS function management
 - Network file system protocols
 - Drive resources that use industry-standard storage protocols



Common NAS Protocols

- Common Internet File System (CIFS) is a public file system used to request and share file transfers. An open, cross-platform technology, CIFS is derived from the Server Message Block (SMB) protocol of Microsoft, in which Windows hosts mount and access shared resources from a CIFS server as if they were locally stored.
- Network File System (NFS), originally developed by Sun Microsystems, enables file sharing across various Unix OSs (including Unix, AIX, and HP-UX) and Unix-like OSs (including Linux and macOS). NFS provides a mount command for mounting remote file systems to the local host, granting access to remotely shared files with the convenience of local file access.



- NFS is a traditional file sharing protocol in Unix environments for independent transmission. It uses TCP or UDP.
- CIFS is a traditional file sharing protocol in Windows environments. It is based on the SMB protocol and usually uses TCP/IP due to its high requirements on network transmission reliability.
- NFS requires dedicated client software, whereas CIFS is integrated into the OS and requires no additional software.
- NFS is a stateless protocol, whereas CIFS is stateful. In the event of a fault, an NFS connection can be automatically restored, while a CIFS connection cannot. Moreover, CIFS sends only a small amount of redundant information, delivering higher transmission efficiency than NFS.

Characteristics of NAS

- Enables data sharing and file-level data access with minimum storage management costs.
- Does not require multiple file servers.
- Alleviates bottlenecks associated with user access to general-purpose servers.
- Uses network and file sharing protocols for archiving and storage.

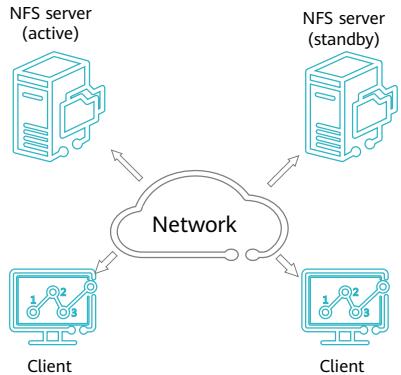
- Advantages of NAS:
 - Scale-out capability: Expanding storage capacity of NAS is as simple as adding drives. New storage can be enabled without network downtime, server upgrade, or server replacement.
 - High performance: With NAS serving as the dedicated file service, other devices on the network are relieved of file sharing responsibilities. NAS networks can be tailored for big data, multimedia storage, and other use cases to enhance performance.
 - Easy setup: Implementing a NAS system can be achieved through a simple script or a device pre-installed with a tailored OS, significantly reducing configuration and management time.
 - Accessibility: All devices on the network can access NAS.
 - Fault tolerance: NAS devices can be formatted to utilize drive cloning, RAID configurations, and erasure coding, thereby ensuring data integrity.

NAS Application Scenarios

- Home storage: home databases of family photos, videos, and other data
- Small-sized enterprise storage: data and file storage, sharing, backup, and recovery
- Remote office: storage, sharing, remote access and management
- Multimedia storage: audio, video, and other multimedia data
- Data backup: data backup and recovery for reliability and security purposes

Introduction to NFS

- NFS allows a computer to mount drive partitions to a remote computer in the same way as mounting local partitions, enabling quick and seamless file sharing over the network.



19 Huawei Confidential



- NFS client applications can transparently read and write files on the remote NFS server as if they were local.
- It enables file sharing between Linux and Unix, but not between Linux and Windows.
- NFS authentication relies solely on IP addresses.

Essential NFS Software

- rpcbind (main program for Remote Procedure Call, or RPC)
 - Also called portmapper, rpcbind enables NFS port mapping and listens on port 111. Run the **rpcinfo** command, for example, **rpcinfo -p localhost**, to check port mappings.
- nfs-utils (main program for NFS)
 - nfs-utils provides the **rpc.nfsd** and **rpc.mountd** daemons, along with related documents, and description and executable files.

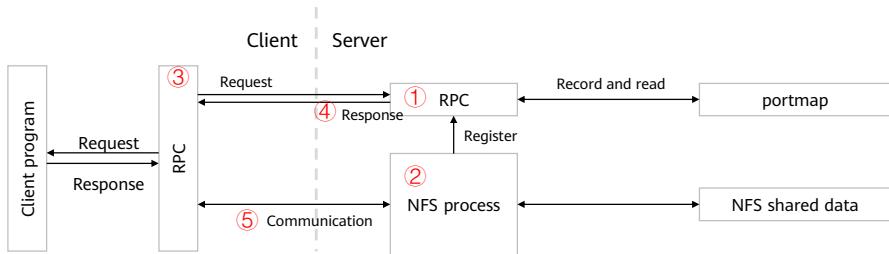
- rpcbind converts RPC program numbers into universal addresses. A host must have a running rpcbind service to initiate RPC requests to a remote host. rpcbind must be started before any RPC management service is started.
- Running **yum install nfs-utils** will also install rpcbind.
- RPC enables programs to request a service from a program located on another computer, abstracting away the network details. RPC presumes the existence of a transport protocol, such as TCP or UDP, for carrying the message data between programs. RPC operates at the transport and application layers of the Open Systems Interconnection (OSI) model.
- RPC follows the client-server model, where the requester acts as a client and the service provider as a server. The client sends parameterized requests to the server and then waits for corresponding responses. The server process sleeps until a request is received, upon which the server extracts the parameters, calculates the results, and sends a response back to the client. It then awaits the next incoming request. The client, upon receiving the response containing the results, proceeds accordingly.
- portmapper (or portmap): Upon startup, an RPC service selects an idle port to listen on (the port may vary each time) and registers itself with portmap as an available service. The RPC server informs the portmap of the port number it is listening on and the corresponding RPC program number it serves. Generally, the

portmapper process uses TCP/UDP port 111. An RPC service corresponds to a unique RPC program number.

- nfsd: This user-mode process of the server redirects requests to the kernel transport endpoint and calls kernel-mode processes to handle those requests and respond to clients.

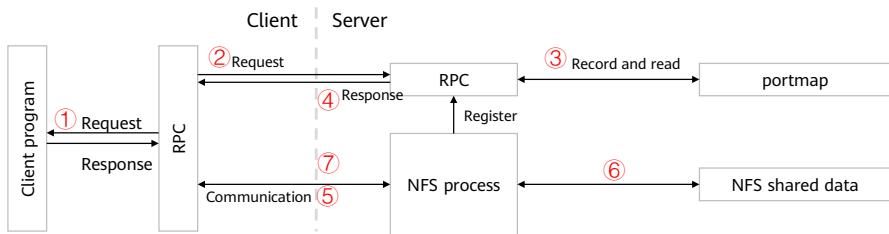
Working Principles of NFS

- NFS server and connection
 - The server starts the RPC service and NFS process, and registers port information with the RPC service.
 - The client starts the RPC service (portmap) and requests the NFS server port number from portmap running on the server, which transmits the information.
 - portmap on the server sends the NFS port information to the client.
 - The client uses the obtained NFS port number to connect to the NFS server for data transmission.



Working Principles of NFS

- NFS user connection and data access
 - When a user accesses a website, the backend program sends an NFS file access request from the NFS client. The client RPC service sends an NFS file query request to port 111 of the server RPC service.
 - The server RPC service locates the registered NFS port and notifies the client RPC service.
 - The NFS client obtains the correct port and connects to the NFS daemon for data access.
 - The NFS client accesses the data and returns it to the frontend program, which then presents the access result to the user, completing the access operation of the website user.



NFS Server Installation

- Check if nfs is installed on the server.
 - rpm -aq | grep nfs
- Check if rpcbind is installed on the server.
 - rpm -aq | grep rpcbind
- Install nfs.
 - yum install nfs-utils
- Enable the NFS and rpcbind services to automatically start upon system startup.
 - systemctl enable nfs --now

NFS Server Configuration

- Create a shared directory: **mkdir /XX/**
- Edit the configuration file: **vi /etc(exports**
 - Add a line: **/XX/ *(rw,sync)**
- Restart the NFS service: **systemctl restart nfs**
- Check whether a shared directory exists: **showmount -e**

| Configuration File Parameter | Description |
|------------------------------|---|
| rw | The directory is accessed in read/write mode. |
| ro | The directory is accessed in read-only mode. |
| sync | Data is synchronously written to the memory and drive. |
| async | Data is temporarily stored in the memory and is not directly written to the drive. |
| no_root_squash | The root user is granted the root permissions after logging in to the NFS server. |
| root_squash | The root user is granted the permissions of anonymous user nobody after logging in to the NFS server. |
| all_squash | Whoever logs in to the NFS server is granted the permissions of anonymous user nobody . |
| insecure | Unauthorized access from the server is allowed. |

- **/XX/** indicates the shared directory on the server.
- ***** indicates that any IP address can access the shared directory.
- If the client IP address is specified, only the host with the specified IP address is allowed to access the directory.
- No space is allowed between ***** or the IP address and the quotation marks.
- **anonuid** sets the user ID for any user who logs in to the NFS server. The ID must exist in **/etc/passwd**.
- **anongid** is similar to **anonuid** but sets the group ID.

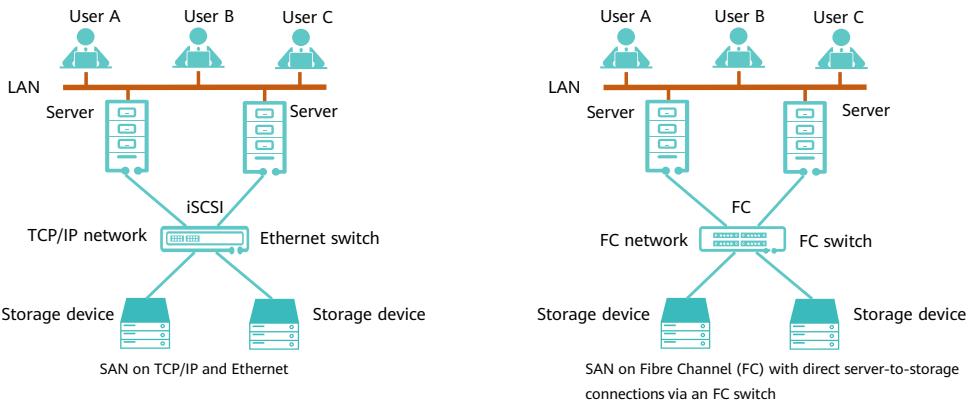
NFS Client Configuration

- Check the mount point.
 - `showmount -e server_IP_address`
- Mount a remote directory.
 - `mount -t nfs server_IP_address:/XX/ /mnt/app/test`
- Check whether the mounting is successfully.
 - `df -h`
- Edit `/etc/fstab` to mount the directory upon system startup.
 - `server_IP_address:/xx/mount_directory nfs defaults,_netdev 0 0`

- `/XX/` is the remote shared directory, which is separated from the IP address by a colon (:).
- `/mnt/app/test/` is the mount point on the client.
- `mount -t` specifies the file system type. This option is not required in most cases. `mount` can automatically select an appropriate type.
- `-t nfs` indicates that the file system type to be mounted is specified.
- Format for configuring automatic mounting: Device or pseudo file system to mount Mount point File system type Mount options Dump frequency Self-check order
- NFS automatic mounting parameters:
 - Device to mount: `server_IP_address:/xx/`
 - Mount point: `/mnt/app/test`
 - File system type: **nfs**
 - Mount options: **defaults**
 - Dump frequency: 0 disables dumps; 1 enables daily dumps; and 2 enables dumps every other day.
 - Self-check order: 0 disables self-check; 1 indicates that this file system is checked first, which is typically used for the root file system.

SAN

- The SAN is a network architecture that connects drive arrays, tape libraries, optical disc libraries, and other devices for efficient data access and management.



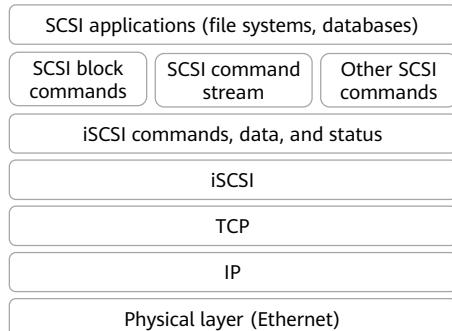
26 Huawei Confidential



- A SAN typically includes:
 - Storage devices connect to the storage switch through FC to provide high-speed data transmission and storage. Examples include
 - Storage switches that, as the core of the SAN, utilize FC to manage and schedule data transmission between storage devices and
 - Hosts that access the storage devices in the SAN though the storage switch.
- SANs are widely used in enterprise data storage and virtualization due to their advantages in terms of high-speed transmission, reliability, management, and scalability.

Introduction to iSCSI

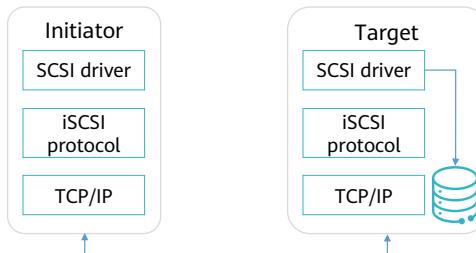
- Internet Small Computer System Interface (iSCSI) is a processing standard for system-level interfaces that connect computers and smart devices. Running on the IP protocol, iSCSI allows users to build SANs on TCP/IP networks.



- iSCSI enables SCSI to run over IP networks. It performs route selection in networks such as high-speed Gigabit Ethernet to achieve connection between SCSI and TCP/IP.
- iSCSI also establishes and manages connections between IP-based storage devices and encapsulates SCSI data for transmission over the existing IP network.

Basic iSCSI Components

- An iSCSI session is established between an initiator and target, which are identified by iSCSI qualified names (IQNs).
- The target is a storage device or a collection of storage devices on the server, identified by a logical unit number (LUN).
- The initiator is a client that connects to the target.



28 Huawei Confidential



- The target can be a drive array or a host equipped with drives. It is a storage device or a collection of storage devices on the server, identified by a LUN. The iSCSI target operates over sockets, listening on a default port of 3260. It utilizes the TCP protocol to ensure data security.
- The initiator is a client tool that connects to the target created on the server. It can discover and use the drive space mapped to the network by the target.
- An iSCSI client must be authenticated using either its IP address or the Challenge Handshake Authentication Protocol (CHAP).

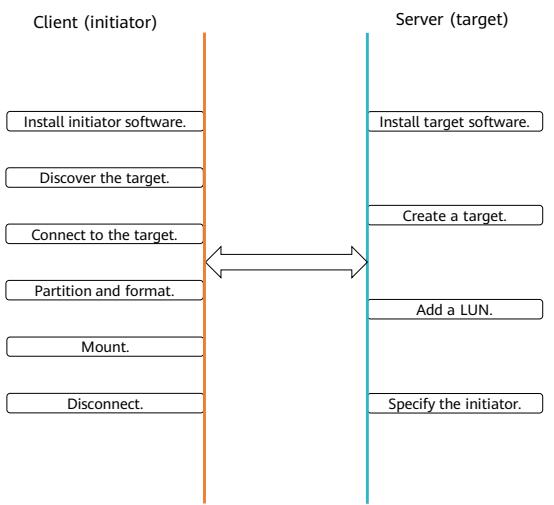
Introduction to IQN

- IQNs are used to identify iSCSI devices.
- IQNs are in the format of **iqn.yyyy-mm.<reverse domain name>[:identifier]**
 - *yyyy-mm* is the year and month.
 - *<reverse domain name>* is globally unique.
 - *identifier* is an identifiable name, which is customizable.
- In openEuler, the IQN configuration of the initiator is stored in **/etc/iscsi/initiatorname.iscsi**.

- The IQN uniquely identifies the target. Clients use the IQN to find the target. IQNs comply with a naming standard, for example, **iqn.20XX-XX.com.example:storage1**.
- Reverse domain name example: **com.test**, which is the reverse of **test.com**

SAN Configuration on Linux

- Configure a target:
 - Install target software.
 - Create a target.
 - Add a LUN to the target.
 - Determine the initiator.
- Configure an initiator:
 - Install initiator software.
 - Discover the target.
 - Connect to the target.
 - Partition and format.



30 Huawei Confidential



- The target can be discovered by the client only after it is configured on the server.
- Install target software: **yum install -y scsi-target-utils**
- Install initiator software: **yum -y install iscsi-initiator-utils**

Target Configuration

- Using the CLI.
 - targetcli: a user-mode CLI provided by the Linux-IO Target (LIO) utility for managing targets.
 - tgtadm: The most popular target management tool prior to LIO, it manages LUNs and ACLs, detects connected initiators, and supports iSNS services.
 - ietadm: creates iSCSI devices using the **iscsi_trgt.ko** kernel module.
- Using the configuration file.
 - Settings made with the **tgtadm** command are not persistent and need to be saved to a configuration file in order for the system to load them upon the next startup.

Introduction to tgtadm

- **tgtadm** changes target configurations temporarily. After the target service is restarted, the configurations become invalid.
- **tgtadm** command syntax: **tgtadm --lld [driver] --op [operation] --mode [mode] [OPTION]....**
- Common **tgtadm** options and values include:
 - **driver** (driver program)
 - iscsi
 - **mode** (operation object)
 - target
 - logicalunit
 - account
 - **OPTION** (operation)
 - delete
 - new
 - bind & unbind
 - show

- **target**: iSCSI target, which is a virtual device to be accessed by multiple initiators
- **logicalunit**: logical storage unit on an iSCSI target, which can be regarded as an individual storage device by an iSCSI initiator
- **account**: user account on an iSCSI target, which is used to authorize iSCSI initiators to access the target or logical unit.

Common tgtadm Options

- Common options

| Option | Long Option | Description |
|--------|---------------------|--|
| -t | --tid | Target ID |
| -T | --targetname | Target name |
| -l | --lun | LUN |
| -b | --backing-store | Backend storage device to associate with the specified LUN |
| -I | --initiator-address | IP address that can access the target |
| -Q | --initiator-name | Client user who can connect to the target |

- iSCSI Extensions for RDMA, or iSER, is an extension of iSCSI, the TCP/IP storage network standard. iSER leverages both the RDMA protocol suite and iSCSI components.

Permanent Target Configuration

- Run the **tgt-admin --dump** command to query current configurations. Copy the output information to the configuration file.
- Add the following lines to the **/etc/tgt/targets.conf** configuration file:

```
<target iqn.2018-09.com.example:target1>
    backing-store    /dev/vdb1      # Specify an existing
                                partition.
    initiator-address 192.168.1.11   # Client access
    control
</target>
```

Target Configuration Example

- Requirement: Create a target named **iqn.2023.04.com.openeuler:target1** in the SAN storage, set **tid** to **1**, add a LUN **/dev/vdb** to the target, and allow access from 192.168.1.1.
- Procedure:

```
[root@web ~]# yum -y install scsi-target-utils
[root@web ~]# tgtadm --lld iscsi --mode target --op new --tid 1 --targetname iqn.2023-04.com.openeuler:target1
[root@web ~]# tgtadm --lld iscsi --mode logicalunit --op new --tid 1 --lun 1 --backing-store /dev/vdb
[root@web ~]# tgtadm --lld iscsi --mode target --op bind --tid 1 --initiator-address 192.168.1.1
[root@web ~]# tgt-admin --dump
default-driver iscsi

<target iqn.2023-04.com.openeuler:target1>
    backing-store /dev/sdb
    initiator-address 192.168.1.1
</target>
[root@web ~]# tgt-admin --dump | grep -A 4 target >> /etc/tgt/targets.conf
```

iscsiadm – Initiator Configuration

- **iscsiadm** is a mode-based tool, whose mode is specified by the **-m** or **--mode** option.
- Syntax:
 - `iscsiadm [-m mode] [-d debug_level] [-P printlevel] [-L all,manual,automatic] [-U all,manual,automatic] [[-T tar-getname -p ip:port -l iface] [-l | -u | -R | -s]] [[-o operation]`

| Option | Description | Available Parameters |
|-----------------|---|--|
| <code>-m</code> | Mode | node, discovery, session, iface |
| <code>-t</code> | Type | sendtargets, slp.fw |
| <code>-p</code> | Portal, that is, the IP address and port number of the server | The default port number is 3260 . |
| <code>-T</code> | Target name | Target IQN |
| <code>-l</code> | Login node server | |
| <code>-d</code> | Debug information level | 0-8 |

- `iscsid`: iSCSI client daemon
- `iscsi`: program that automatically logs in to the target
- `iscsiadm`: client command for managing targets

Initiator Configuration Example

- Requirement: The client scans for and discovers targets on server 192.168.1.2 and logs in to target **iqn.2023-04.com.openeuler:target1**.
- Procedure:
 - Configure the client to discover the target.
 - `iscsiadm -m discovery -t sendtargets -p 192.168.1.2`
 - Connect the client to the target.
 - `iscsiadm -m node -T iqn.2023-04.com.openeuler:target1 -p 192.168.1.2 -l`

Contents

1. Basic Storage Concepts
2. Storage Types
3. Introduction to Shared Storage
- 4. Introduction to Distributed Storage**

Common Distributed Storage Architectures

- Hadoop Distributed File System (HDFS)
 - HDFS, a core component of Hadoop, is a distributed file system designed to run on general-purpose hardware. It provides the basis for distributed computing storage and management.
- Swift
 - Swift is a distributed object storage service initially developed by Rackspace and donated to the OpenStack open source community in 2010.
- Ceph
 - Ceph is a unified storage platform providing block, object, and file storage.
 - Initially developed by Sage Weil during his PhD research, Ceph was first released in 2004 and later donated to the open source community.
- GlusterFS
 - GlusterFS is an open source distributed file system that aggregates storage resources across multiple servers into a virtual file system, enhancing reliability, scalability, performance, and fault tolerance.

- HDFS: centralized control node architecture
 - Client: interacts with HDFS. It splits files to upload into blocks, interacts with the NameNode to obtain file information, and interacts with DataNodes to access data.
 - NameNode: **primary node of an HDFS cluster that maintains the file system tree, including all files and directories in the tree.**
 - DataNode: performs operations and stores actual data blocks.
 - Secondary NameNode: assists the NameNode.
- Swift: decentralized architecture based on consistent hashing
 - Swift leverages a fully symmetric, resource-oriented, distributed architecture, where all components are scalable and single points of failure are avoided.
- Ceph: decentralized architecture based on calculation
 - The base storage system of Ceph, the Reliable Autonomic Distributed Object Store (RADOS), ensures high reliability and scalability. A Ceph cluster comprises Ceph Object Storage Daemons (OSDs) and Ceph Monitors

(MONs).

- The librados basic library abstracts and encapsulates RADOS and provides an API for the upper layer.
- Upper-layer application interfaces include the RADOS Gateway (radosgw), RADOS block devices (RBDs), and Ceph File System (CephFS).
- The application layer contains applications that leverage Ceph interfaces.

- Ceph Functional Modules
 - Client: responsible for storage protocol access and node load balancing
 - MON: monitors the entire cluster, maintains cluster health, and preserves charts detailing cluster status, such as the OSD map, monitor map, placement group (PG) map, and CRUSH map.
 - Metadata Server (MDS): stores metadata of the file system and manages the directory hierarchy.
 - OSD: stores, replicates, balances, and restores data, and checks heartbeats of other OSDs. Typically, one drive corresponds to an OSD.
- Other distributed storage:
 - Lustre is an open source, scalable, and high-performance object-oriented parallel file system. It is widely used in high-performance computing (HPC) on supercomputing platforms.

GlusterFS Advantages

- Petabyte-scale storage
- Support for thousands of clients
- POSIX compatibility
- General-purpose hardware
- Open source
- Compatibility with any drive file system that supports extended attributes
- Support for industry standard protocols, such as NFS and SMB
- Support for replication, quota, geo-replication, snapshot, and BitRot detection
- Optimization techniques for different workloads

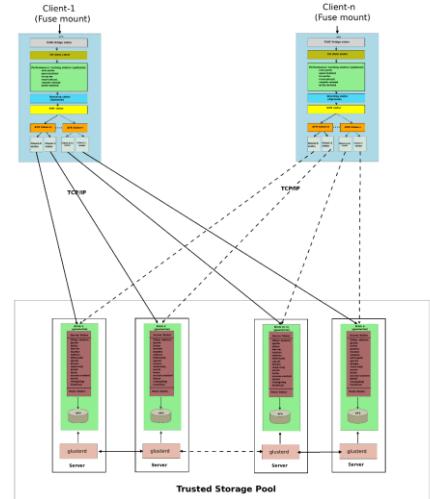
GlusterFS Features

- **High availability:** Automatically detects storage node faults and migrates data to healthy nodes.
- **Load balancing:** Distributes data evenly across multiple storage nodes.
- **Scalability:** GlusterFS systems can be scaled out by adding storage nodes, accommodating growing storage demands.
- **Data security:** Replicates data across nodes for redundancy purposes.
- **High performance:** Uses an elastic hash algorithm to enable rapid, concurrent read and write operations by multiple clients.
- **Easy management:** Provides a comprehensive toolset for streamlined management of storage nodes and file systems.

- The elastic hash algorithm of GlusterFS eliminates the need for MDSs, thereby avoiding single points of failure and performance bottlenecks, and enabling truly parallel data access.
 - The elastic hash algorithm is an implementation of the Davies-Meyer algorithm, which produces a 32-bit integer hash value.
 - Assuming a logical volume comprises N bricks (storage units), this algorithm divides the 32-bit integer range into N consecutive subspaces, each corresponding to a brick.
 - When a user or application accesses a namespace, the algorithm calculates a hash value for that namespace. This hash value determines the corresponding 32-bit integer space, which then identifies the specific brick storing the data.
- GlusterFS is compatible with all storage types. It does not rely on proprietary data formats and uses mainstream file systems like ext3 and XFS. GlusterFS supports NFS, CIFS, HTTP, FTP, SMB, and its native protocol. They are all compatible with POSIX standards.

GlusterFS Architecture

- Brick (storage block):
 - The basic storage unit in GlusterFS. It represents an exported directory on a server in the trusted storage pool.
- Volume (logic volume):
 - Collection of bricks. Most management operations are performed on volumes.
- Filesystem in Userspace (FUSE):
 - A kernel module that allows users to create their own file systems without modifying the kernel.
- Virtual File System (VFS):
 - Interface provided by the kernel for drive access.
- glusterd (management daemon):
 - Manages processes running on each node in the storage cluster



- Source: <https://docs.gluster.org/en/latest/Quick-Start-Guide/Architecture/>

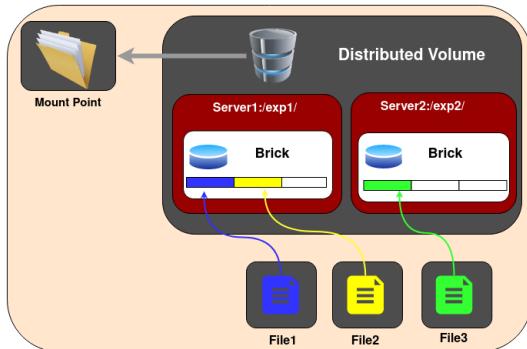
Common GlusterFS Volume Types

- Distributed volume
- Replicated volume
- Distributed replicated volume
- Dispersed volume
- Distributed dispersed volume

- There are seven volume types in GlusterFS:
 - Distributed volume is the default GlusterFS volume type. It distributes files across all brick servers using a hash algorithm, where each file is directly stored on a server node, without being split, similar to RAID0 for files.
 - Dispersed volumes, also called erasure coded volumes or dispersed redundant volumes, splits files into data blocks, distributes them across multiple brick servers in round-robin mode. Files are stored as data blocks without redundancy, similar to RAID5 or RAID6.
 - Replicated volumes synchronize files to multiple brick servers as replicas, similar to RAID1.
 - Distributed dispersed volume means the number of brick servers is a multiple of the total stripes. A distributed striped volume comprises at least four servers, similar to RAID 50.
 - Distributed replicated volume means the number of brick servers is a multiple of the total replicas. Distributed replicated volumes ensure redundancy in a way similar to RAID10.

Distributed Volumes of GlusterFS

- Files are distributed on bricks in a volume. **File1** may be stored on either brick 1 or brick 2, but not both. Distributed volumes do not have data redundancy.



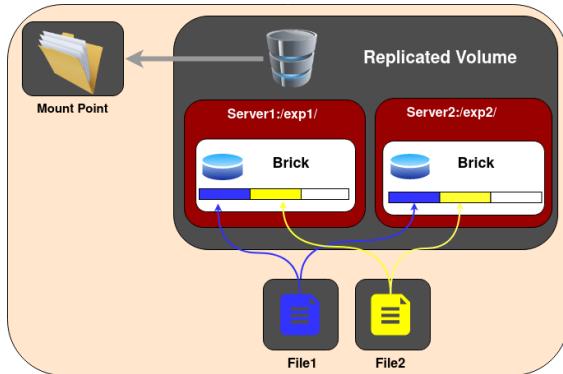
45 Huawei Confidential



- Such volumes offer simple and cost-effective capacity expansion, but are susceptible to complete data loss in the event of a brick failure. The underlying hardware must provide data protection.
- **File1** and **File2** are stored on **Server1**, and **File3** is stored on **Server2**. Files are stored randomly. A file (for example, **File1**) can be stored on either **Server1** or **Server2**, but not both **Server1** and **Server2**.
- Suitable for storage of non-critical data, such as logs and caches.
- The following command creates distributed volume **dis-volume** and distributes files to **server1:/exp1**, **server2:/exp2**, and **server3:/exp3** based on file hashes.
- `gluster volume create dis-volume server1:/esp1 server2:/exp2 server3:/exp3`
- Images on this page and the following four pages are from [Architecture - Gluster Docs](#).

Replicated Volumes of GlusterFS

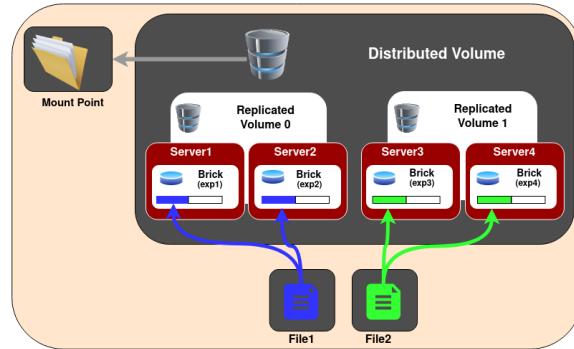
- Replicated volumes ensure data redundancy by storing data replicas across all bricks. When a volume is created, a client can specify the number of replicas within a volume for reliability and data redundancy.



- Suitable for storage of critical data, such as databases and file systems.
- File1** exists on both **Server1** and **Server2**. The same applies to **File2**. Files on **Server2** are equivalent to replicas of files on **Server1**.
- A volume with two replicas requires at least two bricks. A volume with three replicas requires at least three bricks.
- Each server in the volume stores a complete replica for redundancy.
- Within a volume, the number of replicas can be customized but must be the same as the number of storage servers (bricks).
- There is a high risk of split brain when there are at least two servers with two replicas. You are advised to deploy at least three replicas.
- If nodes within a volume have varying storage capacities, the total capacity of the volume is limited to that of the node with the least space (law of the minimum).

Distributed Replicated Volumes of GlusterFS

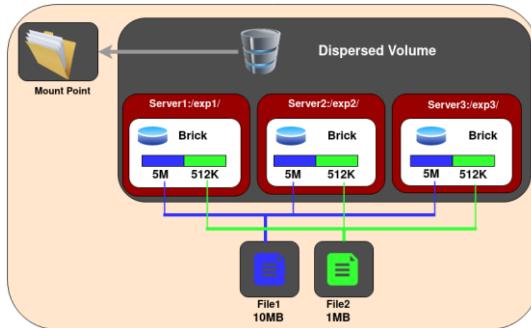
- Files are distributed across sets of replicated bricks. The number of bricks must be a multiple of the number of replicas. It is worth noting that the order in which bricks are specified is important because consecutive bricks become replicas of each other.



- In a volume with eight bricks and two replicas, the first two bricks, and each following pair, become replicas of each other. This volume is denoted as a 4x2 volume. Similarly, in a volume with eight bricks and four replicas, four bricks become replicas of each other, and this volume is denoted as a 2x4 volume.
- **File1** and **File2** are located to **Server1** and **Server3** respectively by the distributed volume. To store **File1**, two replicas (**exp1** on **Server1** and **exp2** on **Server2**) are created due to the nature of replicated volumes. Similarly, to store **File2**, two replicas (**exp3** on **Server3** and **exp4** on **Server4**) are created.
- This suits critical storage (such as databases and file systems) that have demanding read/write performance requirements.

Dispersed Volumes of GlusterFS

- Dispersed volumes are based on erasure codes. They stripe the encoded data of files, add some redundancy, and store them across multiple bricks in the volume. Dispersed volumes are highly reliable and do not waste space. When a volume is created, a client can specify the number of redundant bricks within a volume.



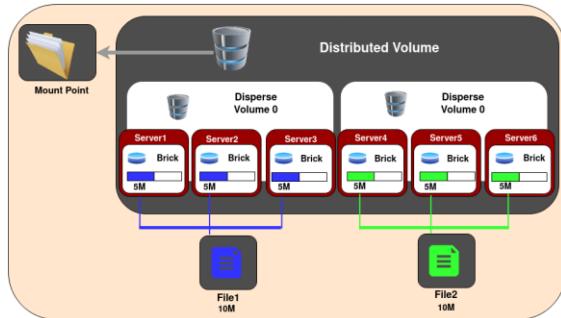
48 Huawei Confidential



- The number of redundant bricks determines how many bricks can be lost without interrupting volume operations.
- Dispersed volumes are also called erasure coded volumes or dispersed redundant volumes.
- Provides high-availability storage protection for important data, databases, and VM images.
- The dispersed volume splits the 10 MB **File1** into two data blocks, encodes them into three blocks using erasure coding, then distributes the encoded blocks across physical nodes on **Server1**, **Server2**, and **Server3**. When data is needed, the dispersed volume retrieves the encoded blocks, decodes them back into the original data blocks using erasure coding, combines them, and returns the data to the user. **File2** undergoes the same process.

Distributed Dispersed Volumes of GlusterFS

- Distributed dispersed volumes are the same as distributed replicated volumes, but use dispersed subvolumes instead of replicated ones. The number of Bricks must be a multiple of that of the first subvolume. The purpose for distributed dispersed volumes is to easily scale the volume size and distribute the load across bricks.



49 Huawei Confidential



- Erasure coded volumes and distributed erasure coded volumes offer high availability and data protection. Of the two, distributed erasure coded volumes are better suited for dealing with large-scale data storage and applications that demand high performance.

GlusterFS Installation and Deployment Procedures

- Install the server.
 - Prepare at least three nodes.
 - Format and mount the bricks.
 - Install glusterfs-server.
 - Configure the firewall.
 - Configure the trusted pool.
 - Create and enable a GlusterFS volume.
- Install the client.
 - Configure the firewall.
 - Install the client software.
 - Add host name resolution configurations for the nodes.
 - Mount a GlusterFS volume.

GlusterFS Installation and Deployment – Brick Formatting and Mounting

- Format and mount bricks on all nodes.
- It is recommended that the storage capacities of bricks within the same volume be the same. If the capacities are different, the minimum capacity takes effect.
- A brick can be a directory of any file system.
- The specific configuration procedure is as follows:

```
mkfs.xfs -i size=512 /dev/sdb1      ##### Create partition sdb1.  
mkdir -p /data/brick1                ##### Create a directory and use it as a brick.  
echo '/dev/sdb1 /data/brick1 xfs defaults 1 2' >> /etc/fstab    ##### Set partition sdb1 to be automatically  
mounted to the brick upon system startup.  
mount -a && mount
```

GlusterFS Installation and Deployment – Volume Creation

- Commands for creating GlusterFS volumes:
 - Distributed volume:
 - `gluster volume create <volume-name> [transport [tcp | rdma | tcp,rdma]] brick1,brick2...`
 - Replicated volume:
 - `gluster volume create <volume-name> [replica <COUNT>] [transport [tcp | rdma | tcp,rdma]] brick1,brick2...`
 - Distributed replicated volume:
 - `gluster volume create <volume-name> [replica <COUNT>] [transport [tcp | rdma | tcp,rdma]] brick1,brick2...`
 - Dispersed volume:
 - `gluster volume create <volume-name> [disperse [<COUNT>]] [disperse-data <COUNT>] [redundancy <COUNT>] [transport tcp | rdma | tcp,rdma] brick1, brick2...`
 - Distributed dispersed volume:
 - `gluster volume create <volume-name> [disperse [<COUNT>]] [disperse-data <COUNT>] [redundancy <COUNT>] [transport tcp | rdma | tcp,rdma] brick1, brick2...`

GlusterFS Installation and Deployment – Volume Creation

- Command options in the volume creation commands:
 - *<volume-name>*: volume name
 - **transport**: transmission mode, which can be **tcp**, **rdma**, **tcp**, or **rdma**
 - *<COUNT>*: quantity
 - *brick1,brick2...*: bricks used to form the volume. You can add multiple bricks and separate them with spaces. The format is *nodeX/XX*.
 - **disperse-data**: quantity of slices each file in a dispersed volume is divided into before erasure coding is applied
 - **disperse**: quantity of slices each file in a dispersed volume is eventually divided into, that is, how many bricks the file is dispersed into
 - **redundancy**: number of redundancy slices in a dispersed volume, that is, extra slices after erasure encoding is applied

GlusterFS Installation and Deployment – Firewall and Trusted Pool Configuration

- Firewall configuration:
 - `iptables -I INPUT -p all -s <other-ip1-address> -j ACCEPT`
- Trusted pool configuration:
 - A trusted pool is a trusted server pool (TSP) consisting of all GlusterFS service nodes. A TSP can consist of just one node.
 - Add a node to a trusted pool (the node that executes the command does not need to be in a pool): **gluster peer probe nodeX**
 - Check the trusted pool status: **gluster peer status**

- The Gluster processes on each node need to access and communicate with each other. In a secure network, you can open all ports on each node to the others.
- An arbitrary number of drive bricks in the trusted pool can be combined to form a large storage volume.
- Once a trusted storage pool is established, only existing members can add new nodes. New nodes cannot detect the pool and must be detected by a pool member.

GlusterFS Client Configuration – Volume Mounting

- Three mounting options are available for GlusterFS volumes:
 - Gluster Native Client: Specify the **glusterfs** parameter when mounting a volume on a client. Gluster Native Client supports high availability.
 - `mount -t glusterfs nodeX:NEW-VOLNAME /mnt/XX`
 - NFSv3: Enable NFS for a volume and restart the volume. After the volume is restarted, the server automatically starts the NFS process to allow mounting via NFS.
 - `gluster volume set VOLNAME nfs.disable off`
 - `mount -t nfs nodeX:NEW-VOLNAME /mnt/XX`
 - Samba: On the server, ensure that **nfs.disable** for the volume is **on**, configure volume options, and restart the volume and glusterd service. On the client, install and start Samba and open related ports.

- Additional steps are needed before you can mount volumes using Samba. On the server, create a Samba user with the permission to modify the directory to mount. On the client, Install samba-client and cifs-utils.
 - On the server:
 - Add **option rpc-auth-allow-insecure on** to the **/etc/glusterfs/glusterd.vol** file.
 - `gluster volume set VOLNAME server.allow-insecure on`
 - `gluster volume set VOLNAME storage.batch-fsync-delay-usec 0`
 - `gluster volume set VOLNAME stat-prefetch off`
 - `adduser -s /sbin/nologin smbuser`
 - `smbpasswd -a smbuser`
 - `mount -t glusterfs nodeX:VOLNAME /mnt`
 - On the client:
 - `yum install -y cifs-utils`
 - `yum install -y samba-client*`
 - `smbclient -L nodeX -U% # Verify the Samba shared directory.`
 - `mount -t cifs -o user=smbuser,pass=XXX //nodeX/VOLNAME /mnt #`

Mount the directory as smbuser.

Gluster Client Deployment

- Install the client service:
 - `yum install -y glusterfs glusterfs-fuse`
- Add address resolution to all storage nodes on the client.
- Mount the volume to a created mount point:
 - `mount -t glusterfs nodeX:NEW-VOLNAME /mnt/XX`
- Edit **/etc/fstab** to mount the volume upon system startup:
 - `nodeX:NEW-VOLNAME /mnt/XX glusterfs defaults,_netdev 0 0`

- Format for configuring automatic mounting:
*HOSTNAME_OR_IPADDRESS:VOLNAME MOUNTDIR glusterfs defaults,_netdev
0 0*

GlusterFS Volume Management – Starting, Stopping, Viewing, and Rebalancing

- Start a volume:
 - gluster volume start *VOLNAME*
- Stop a volume:
 - gluster volume stop *VOLNAME*
- Check volume information:
 - gluster volume status *VOLNAME* # Check the status of a volume.
 - gluster volume info # Check volume information.
- Rebalance a volume:
 - gluster volume rebalance *VOLNAME start*
 - gluster volume rebalance *VOLNAME status*

GlusterFS Volume Management – Deletion

- Stop a volume:
 - gluster volume stop VOLNAME
- Delete the volume:
 - gluster volume delete VOLNAME
- Bricks released by deleting a volume cannot be used immediately. To use such a brick, delete the **.glusterfs** file from the brick directory and run the following commands:
 - setfattr -x trusted.glusterfs.volume-id /BRICK
 - setfattr -x trusted.gfid /BRICK

- **setfattr -x trusted.glusterfs.volume-id** deletes extended attribute **trusted.glusterfs.volume-id** of a file or directory. This GlusterFS extended attribute identifies the GlusterFS volume to which the file belongs.
- **setfattr -x trusted.gfid** deletes the Global File Identifier (GFID) attribute of a file or directory. A GFID uniquely identifies a file or directory in GlusterFS. The command removes the GFID attribute but leaves the file or directory intact.

GlusterFS Volume Management – Scaling

- Online scaling
 - Scale out a volume online:
 - `gluster volume add-brick VOLNAME NEW-BRICK`
 - Scale in a volume online:
 - `gluster volume remove-brick VOLNAME BRICKNAME start` # Delete the brick.
 - `gluster volume remove-brick VOLNAME BRICKNAME status` # Check the operating status.
 - `gluster volume remove-brick VOLNAME BRICKNAME commit` # Submit the scale-in operation when the status is completed.

- A brick added to a GlusterFS volume can be used only after the volume is rebalanced.
 - `gluster volume rebalance VOLNAME start`
 - `gluster volume rebalance VOLNAME status`
- After a volume is scaled out to prevent excessive load old nodes, manual rebalancing is needed to maintain performance, reliability, and availability. When a volume is scaled in, the system automatically performs rebalancing to prevent data loss caused by node or subvolume removal.

GlusterFS Volume Management – Brick Replacement

- On the server, check the process ID (PID) of the faulty brick process and terminate the process.
 - kill -15 PID
- On the client, unmount the volume that contains the faulty brick.
- On the client, mount the volume to new mount point <CLIENT_MOUNT_POINT>.
- Query the extended attributes of the directory on the replica node of the faulty node: `getfattr -d -m . -e hex BRICK`
- On the client, create a directory, then delete it:
 - cd <CLIENT_MOUNT_POINT>
 - mkdir testDir
 - rm -rf testDir
- Set an extended Attribute to trigger self-healing:
 - setfattr -n trusted.non-existent-key -v abc <CLIENT_MOUNT_POINT>
 - setfattr -x trusted.non-existent-key <CLIENT_MOUNT_POINT>
- Forcibly replace the volume:
 - gluster volume replace-brick VOLNAME BAD-BRICK NEW-BRICK commit force

- Extended attributes (xattrs) are widely supported by modern file systems yet often unnoticed by users. GlusterFS technologies such as Distributed Hash Table (DHT), Automatic File Replication (AFR), and striping extensively utilize xattrs, which consist of key-value pairs with character string keys and binary values.
- In GlusterFS, when an xattr is set or retrieved, related operations are triggered on the Gluster server.
- Each time a brick is added or deleted, the hash range stored in the xattr is recalculated.
- For example, AFR volume **test1** consists of bricks **test1-client-0** and **test1-client-1**. In this case, files in **test1-client-0** have the **trusted.afr.test1-client-1** xattr. Operations on brick are recorded in xattrs in another brick.

Summary

- This course describes the basic concepts and common types of storage, shared storage NAS and SAN, and GlusterFS distributed storage.
- This course aims to prepare you with basic storage knowledge, including various storage architectures and their usage.

Quiz

1. Assume a device cannot be directly used after being mounted to a client, and requires formatting before reading and writing. In this case, which storage type is the most likely? (Single answer)
 - A. Block storage
 - B. File storage
 - C. Object storage
 - D. Backup storage
2. A distributed replicated volume of GlusterFS consists of at least six bricks, with every three bricks forming a replication group. Files within the volume are sliced and distributed across the replication groups. (True or false)
 - A. True
 - B. False

- Answer:
 - A. After a block storage device is mounted, it can be read and written only after being partitioned and formatted, similar to local storage.
 - B. False. A distributed replicated volume consists of at least four bricks, with each pair of adjacent bricks forming a replication group. Files are stored as a whole within different replication groups without being sliced.

Acronyms

| Acronym | Full Spelling |
|---------|--|
| SNIA | Storage Networking Industry Association |
| DAS | Direct-attached storage |
| NAS | Network-attached storage |
| SAN | Storage area network |
| FC | Fiber Channel |
| SCSI | Small Computer System Interface |
| iSCSI | Internet Small Computer System Interface |
| NFS | Network File System |
| CIFS | Common Internet File System |

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and
organization for a fully connected,
intelligent world.

Copyright©2024 Huawei Technologies Co., Ltd.
All Rights Reserved.

The information in this document may contain predictive
statements including, without limitation, statements regarding
the future financial and operating results, future product
portfolio, new technology, etc. There are a number of factors that
could cause actual results and developments to differ materially
from those expressed or implied in the predictive statements.
Therefore, such information is provided for reference purpose
only and constitutes neither an offer nor an acceptance. Huawei
may change the information at any time without notice.



Introduction to openEuler Automated Management



Foreword

- For deployments where many servers need to be rolled out and maintained at the same time, automated operation and maintenance (O&M) is essential. This course introduces two mainstream automated O&M tools, Ansible and SaltStack.

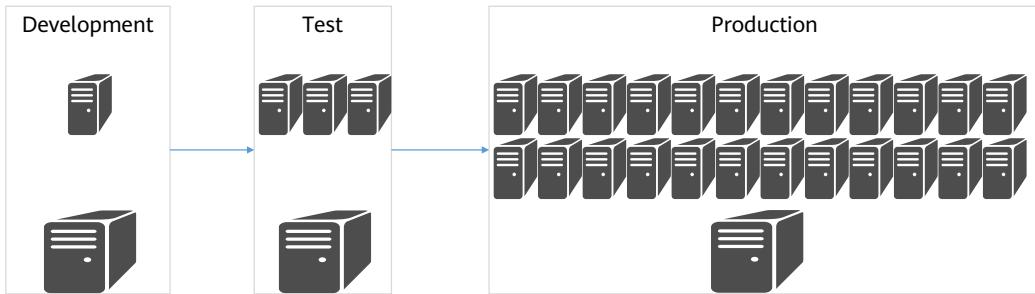
Objectives

- On completion of this course, you will understand:
 - Common Ansible function modules.
 - Development of Ansible playbook.
 - SaltStack remote management.
 - SaltStack configuration management.

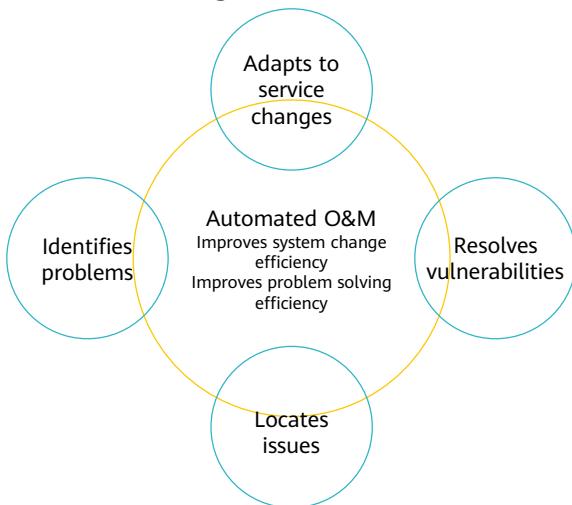
Contents

- 1. Introduction to Automated O&M**
2. Introduction to Ansible
3. Introduction to SaltStack

Sharp Increase in O&M Requests Throughout the Service Lifecycle



Automated O&M Advantages



Automated O&M Scenarios

- Application deployment
- File transfer
- Configuration management
- Task flow orchestration

Common Automated O&M Tools

- Ansible
 - A Python offering for small- and medium-sized environments, it features agentless deployment and client management via SSH.
- SaltStack
 - Also developed with Python, it requires agent deployment on the client to realize efficient execution.

- Generally, Ansible is used when there are fewer than 1,000 servers, whereas SaltStack is recommended when the number of servers exceeds 1,000.

Contents

1. Introduction to Automated O&M

2. Introduction to Ansible

- Ansible Overview and Basic Operations
 - Common Ansible Modules
 - Playbook Overview

3. Introduction to SaltStack

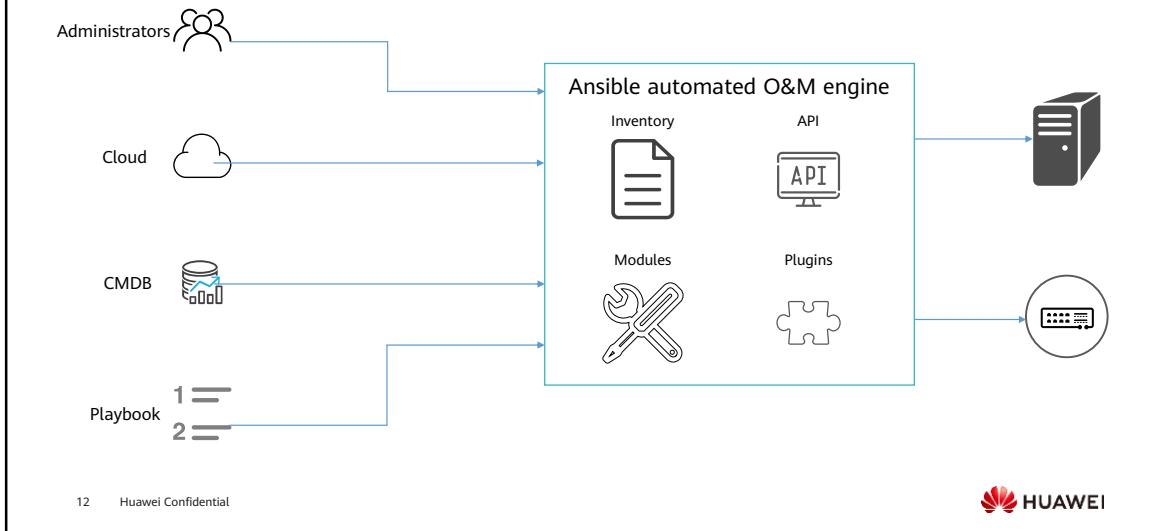
Introduction to Ansible

- The Ansible automation platform provides enterprises a framework for building and operating IT automation at scale. It supports hybrid cloud environments and edge deployments and lets enterprise-wide users (development, O&M, network, and security) create, share, and manage automation tasks.
- Ansible was developed by Michael DeHaan in 2012, and acquired by Red Hat in 2015.
- Paramiko, PyYAML, and Jinja2 are its three key modules.

Ansible Functions and Advantages

- Automation execution environment
 - A container image that executes Ansible playbooks and role-based access control. It provides a defined, consistent, and portable way to build and distribute your environment.
- Automation standardization and scaling up
 - Standardized deployment, startup, authorization, and review. Automation controllers are used to define, manipulate, and extend automation, in which enterprise users can manage inventories, start and schedule workflows, track changes, and integrate reports on centralized UI and REST APIs. Automation controllers support a topology that visualizes multiple sites, including jump, execution, mixing, and control nodes.
- Automation mesh
 - A simple, flexible, and reliable way to extend the automation of large inventories across multiple network topologies, platforms, and regions. Automated mesh provides flexible design options to develop elastic and fault-tolerant architectures, while enhancing security to standardize automation of entire IT assets from core data centers to edges.

Ansible Components and Architecture



12 Huawei Confidential



- Ansible supports Linux, Windows, and various network devices.
- Inventory is a list of objects managed by Ansible. Only objects added to the inventory can be managed by Ansible.
- Ansible has many built-in function modules for administrators to use. Administrators can also customize new function modules.
- Ansible commands can be used to perform operations on objects.
- Playbook is one of the core functions of Ansible. Administrators can write Ansible tasks into YAML files that are automatically executed.

Installation of Ansible

- Installed by Yum

```
yum install -y ansible
```

- Installed from the source code

```
yum install -y python-jinja2 PyYAML python-paramiko python-babel python-crypto  
tar -xvf ansible-xxxxx.tar.gz  
cd ansible-xxxx  
python setup.py build  
python setup.py install
```

- Installed via the Git command

```
git clone git://github.com/ansible/ansible.git --recursive  
cd ./ansible  
source ./hacking/env-setup
```

- Installed by pip

```
pip install ansible --upgrade
```

Key Items in the Ansible Configuration File

- The default Ansible configuration file is **/etc/ansible/ansible.cfg**.

```
[defaults]
# some basic default values...

#inventory      = /etc/ansible/hosts          ##Host list configuration file, which is used to add managed objects.
#library        = /usr/share/my_modules/       ##Directory where the Ansible library file is stored.
#module_utils   = /usr/share/my_module_utils/  ##Directory where the Ansible module file is stored.
#remote_tmp     = ~/.ansible/tmp              ##Path for storing temporary commands on the remote host.
#local_tmp      = ~/.ansible/tmp              ##Local path for storing temporary commands.
#plugin_filters_cfg = /etc/ansible/plugin_filters.yml ##Plugin related configuration.
#forks          = 5                            ##Number of concurrent connections. The default value is 5. Change the value
based on the controller performance.
#poll_interval  = 15                          ##Interval. The default value is 15 seconds.
#sudo_user      = root                         ##Default privilege escalation user.
#sudo_user      = root                         ##Whether to ask for the privilege escalation password when running the
Ansible command. You are advised to change the value to False.
#ask_pass       = True                         ##Whether to ask for the SSH password when running the Ansible command.
You are advised to change the value to False.
#transport      = smart                        ##Command transmission mode.
#remote_port    = 22                          ##Port number for connecting to the managed object.
host_key_checking = False                     ##Whether to verify the SSH key. You are advised to set the value to False.
....
```

Ansible Host Inventory Configuration File

- The Ansible host inventory can be written in multiple ways. Three ways are provided officially.

- Ungrouped hosts can be directly placed in the **hosts** file.

```
# Ex 1: Ungrouped hosts, specify before any group headers.  
green.example.com  
blue.example.com  
192.168.100.1  
192.168.100.10
```

- Hosts in a group are labeled with the group name, followed by the host information.

```
#Ex 2: A collection of hosts belonging to the 'webservers' group  
[webservers]  
alpha.example.org  
beta.example.org  
192.168.1.100  
192.168.1.110
```

- If hosts in the same group comply with a certain rule, they can be combined into one item.

```
#Ex 3: A collection of database servers in the 'dbservers' group  
[dbserver]  
db-[99:101]-node.example.com
```

Common Ansible commands

- **ansible**
 - Main command, which is used by the administrator to run temporary commands.
- **ansible-doc**
 - displays configuration documents.
- **ansible-playbook**
 - orchestrates playbooks and customizes automation tasks.
- **ansible-pull**
 - remotely executes commands.
- **ansible-vault**
 - encrypts files.
- **ansible-console**
 - is a console-based interactive command execution tool.

Basic Usage of Ansible

- Basic Ansible syntax format:

```
ansible <host-pattern> [-m module_name] [-a args]
```

- Common Ansible parameters:

- **-v**: displays the running process. **-vv** or **-vvv**: displays more detailed running process.
- **--list-hosts**: displays the host list, which can be abbreviated as **--list**.
- **-k**: enters the SSH password in interactive mode.
- **-C**: checks but does not execute.
- **-T**: sets the command timeout period.
- **-u**: sets the user who runs the remote command.
- **-K**: enters the sudo password in interactive mode.

Logical Relationship of Ansible

- Logical OR

```
## Host 192.168.1.10 or 192.168.1.11  
ansible "192.168.1.10:192.168.1.11" -m ping
```

- Logical AND

```
##Hosts shared by Zabbix and Nginx host groups  
ansible "Zabbix:&Nginx" -m ping
```

- Logic NOT

```
##Hosts that do not belong to the Nginx host group  
ansible ':!Nginx' -m ping
```

- Regular expression

```
##Hosts on the 192.168.1.0 network segment  
ansible '192.168.1.*' -m ping
```

- A logical relationship can be used multiple times. For example, you can use **A:&B:!C** to view the hosts that are shared by host groups A and B but do not belong to host group C.

Contents

1. Introduction to Automated O&M

2. Introduction to Ansible

- Ansible Overview and Basic Operations
- **Common Ansible Modules**
 - Playbook Overview

3. Introduction to SaltStack

Common Ansible Function Modules – command

- The command module is the default module of Ansible. It is used to remotely run commands on controlled objects. The following special characters should be avoided: <>)
- Common options of this module:
 - **chdir**: enters the specified directory before running the command.
 - **cmd**: indicates the command to be executed.
 - **creates**: determines to run a command based on the status of a file. If there is a file, this command is not executed.
 - **removes**: is contrary to **creates**. It checks whether to run the command based on whether the file exists. If there is a file, this command is executed.
 - **warn**: enables the warning function.

Practice of Using Common Ansible Function Modules – command

- Requirement: Check whether there is the `/data` directory on the host whose IP address is 192.168.1.4. If the directory does not exist, create it, copy `/etc/resolv.conf` to `/data`, and check whether the operation is successful.

```
[root@Ansible ~]# ansible 192.168.1.4 -m command -a "chdir=/ ls"
192.168.1.4 | CHANGED | rc=0 >>
afs
bin
Boot
.....
[root@Ansible ~]# ansible 192.168.1.4 -m command -a "chdir=/ mkdir data"
[WARNING]: Consider using the file module with state=directory rather than running 'mkdir'. If you need to use command because file is insufficient you can add 'warn: false' to this command task or set 'command_warnings=False' in ansible.cfg to get rid of this message.
192.168.1.4 | CHANGED | rc=0 >>
[root@Ansible ~]# ansible 192.168.1.4 -m command -a "cp /etc/resolv.conf /data"
192.168.1.4 | CHANGED | rc=0 >>
[root@Ansible ~]# ansible 192.168.1.4 -m command -a "creates=/data/resolv.conf cat /data/resolv.conf"
192.168.1.4 | SUCCESS | rc=0 >>
skipped, since /data/resolv.conf exists
[root@Ansible ~]# ansible 192.168.1.4 -m command -a "removes=/data/resolv.conf cat /data/resolv.conf"
192.168.1.4 | CHANGED | rc=0 >>
# Generated by NetworkManager
nameserver 192.168.1.1
```

- The execution process in the slide is to show the usage of each parameter. Such operations are not required in actual work.
- When the **mkdir** command is used to create a directory, the system generates a warning because Ansible has a dedicated module to create a directory.

Common Ansible Function Modules – shell and Related Cases

- The shell module is similar to the command module, but it supports special characters: <>|
- The shell module cannot process complex commands, for example, `cat /etc/passwd | awk '{print $1}' > /dev/null`.
- Requirement: Create user `test` on the host whose IP address is 192.168.1.4 and set the password to `openEuler@123`.

```
[root@Ansible ~]# ansible 192.168.1.4 -a "adduser test"
192.168.1.4 | CHANGED | rc=0 >>

[root@Ansible ~]# ansible 192.168.1.4 -m shell -a "echo openEuler@123 | passwd --stdin test"
192.168.1.4 | CHANGED | rc=0 >>
Changing password for user test.
passwd: all authentication tokens updated successfully.
```

Common Ansible Function Modules – script and Related Cases

- The script module is used to send scripts on the controller to the controlled host for execution.
- Requirement: Query the IP address of the VM whose host name is Zabbix.

```
#####Create the following shell scripts on the control node:  
[root@Ansible ~]# cat host-ip.sh  
#!/bin/bash  
if [ $HOSTNAME = "Zabbix" ]; then  
    hostname -i  
fi  
[root@Ansible ~]# ansible all -m script -a "~/host-ip.sh" | grep "stdout"  
"stdout": "",  
"stdout_lines": []  
"stdout": "192.168.1.4\r\n",  
"stdout_lines": [  
"stdout": "",  
"stdout_lines": []
```

Common Ansible Function Modules – copy and fetch

- The copy and fetch modules are used to transmit data between the controller and the controlled host.
- copy is used to copy controller data to the controlled host and to transfer directories; fetch is used to copy and only transfer files from the controlled host to the controller.
- Common options are as follows:
 - **src**: specifies the source file. If the directory is empty, the source file is not copied.
 - **dest**: specifies the destination directory.
 - **content**: is the copy module parameter and cannot be used together with **src**. This option is used to copy the content to the file of the controlled host.
 - **owner**, **group**, or **mode**: are the copy module parameters and specify the owner, group, and permission of the destination file.
 - **force**: is the copy module parameter. If the content of the source file is inconsistent with that of the destination file during replication, set **force** to **yes** to overwrite the destination file. If **force** is set to **no**, the destination file is not overwritten. If the **force** option is not used, the destination file is overwritten.

- The **dest** option of the copy module must be an absolute path. If **src** is a directory, **dest** must also be a directory. If there is no path specified by **dest** and ends with a slash (/) or the source is a directory, the path is automatically created. If **dest** is a relative path, the starting directory is determined by the controlled host. If **src** and **dest** are the same file and there is no destination path specified by **dest**, all parent paths in the path will not be created and this operation will fail.
- In regard to the **dest** option of the fetch module, Ansible stores the file in a specified directory and creates a path that starts with the IP address or host name of the controlled host and the **src** path. For example, **src** is **/test**, the IP address of the controlled node is 192.168.1.4, and the destination path is **/tmp**, the final file is saved in **/tmp/192.168.1.4/test** of the controller.

Practice of Using Common Ansible Function Modules – copy and fetch

- Requirement 1: Copy the **root/data/test** file from the controller to **192.168.1.4/tmp**. During the replication, set the owner to **root:root** and permission to **755**, and enter "hello, openEuler1" for **test**.

```
[root@Ansible ~]# mkdir data
[root@Ansible ~]# touch data/test
[root@Ansible ~]# ansible 192.168.1.4 -m copy -a "src=/root/data/ dest=/tmp/data/ owner=root group=root mode=755"
192.168.1.4 | CHANGED => {
.....
[root@Ansible ~]# ansible 192.168.1.4 -m copy -a "content='hello,openEuler1' dest=/tmp/data/test"
192.168.1.4 | CHANGED => {
.....
[root@Ansible ~]# ansible 192.168.1.4 -a "cat /tmp/data/test"
192.168.1.4 | CHANGED | rc=0 >>
hello,openEuler1
```

```
[root@Ansible ~]# ansible 192.168.1.4 -m fetch -a "src=/tmp/data/test dest=/tmp"
[root@Ansible ~]# ls /tmp/192.168.1.4/tmp/data/
test
[root@Ansible ~]# cat /tmp/192.168.1.4/tmp/data/test
hello,openEuler1
```

Common Ansible Function Modules – file

- The file module is used to manage files and file attributes.
- Common options for the file module are as follows:
 - **force**: forcibly creates soft links in two cases. The source file does not exist, but it will be created later; the target soft link already exists, but it is forcibly canceled and a new soft link is created.
 - **owner, group, or mode**: set the owner, group, or permission.
 - **path**: specifies the path of the managed file. This option is mandatory.
 - **recurse**: recursively sets directory attributes.
 - **src or dest**: specifies the paths of the source file or destination file to be linked when **state** is set to **link**.
 - **state**: Important option. There are multiple configurations. Specifically,
 - **touch**: creates a file. If the file or directory already exists, the last modification time of the file or directory is updated.
 - **directory**: creates a directory and has the recursive creation function of **mkdir -p**.
 - **link or hard**: creates soft or hard links.
 - **absent**: deletes a specific directory or remove a link.

Practice of Using Common Ansible Function Modules – file

- Requirement: Create the **/tmp/file/data** directory on the controlled host whose IP address is 192.168.1.4, set **owner** and **group** to **test:test**, and set **mode** 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.

```
[root@Ansible ~]# ansible 192.168.1.4 -m file -a "path=/tmp/file/data owner=test group=test mode=755 state=directory"
192.168.1.4 | CHANGED => {
...
[root@Ansible ~]# ansible 192.168.1.4 -m file -a "path=/tmp/file/data/test state=touch"
192.168.1.4 | CHANGED => {
...
[root@Ansible ~]# ansible 192.168.1.4 -m file -a "src=/tmp/file/data/test dest=/tmp/link state=link"
192.168.1.4 | CHANGED => {
...
[root@Ansible ~]# ansible 192.168.1.4 -m file -a "path=/tmp/file state=absent"
192.168.1.4 | CHANGED => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": true,
  "path": "/tmp/file",
  "state": "absent"
}
[root@Ansible ~]# ansible 192.168.1.4 -m shell -a "ls /tmp | grep file"
192.168.1.4 | FAILED | rc=1 >>
non-zero return code
```

Common Ansible Function Modules – archive and unarchive

- The archive module is used to pack and compress files, and the unarchive module is used to decompress compressed packages.
- Common options for the archive module are as follows:
 - **owner, group, or mode**: sets the owner, group, or permission.
 - **format**: sets the packaging format. The value can be **bz2, gz, tar, xz, or zip**. The default value is **gz**.
 - **path**: specifies the absolute path(s) of the file(s) to be packed.
 - **exclude_path**: excludes files that do not need to be packed.
 - **remove**: deletes the source file after the packing is complete.
- Common options for the unarchive module are as follows:
 - **copy**: If set to **yes**, the decompressed package is from the controller. Otherwise, the decompressed file is on the controlled node. (default value: yes).
 - **src or dest**: sets the source path of the compressed package or the path for storing the decompressed package. **src** specifies a path on the controller, and when **copy** is set to **no**, **src** specifies a path on the controlled host.
 - **mode**: specifies the permission of the decompressed file.
 - **remote_src**: cannot be used together with **copy**. If this option is set to **yes**, the compressed package is from the controlled node. If set to **no**, the compressed package is from the controller.

Practice of Using Common Ansible Function Modules – archive and unarchive

- Requirement: Create the **test** and **test1** files in the **/tmp** directory on the host whose IP address is 192.168.1.4, compress them into **test.zip**, delete **test** and **test1**, and decompress them to the **/tmp/file** directory.

```
[root@Ansible ~]# ansible 192.168.1.4 -m file -a "path=/tmp/test state=touch"
...
[root@Ansible ~]# ansible 192.168.1.4 -m file -a "path=/tmp/test state=touch"
...
[root@Ansible ~]# ansible 192.168.1.4 -m archive -a "path=/tmp/test,/tmp/test1 format=zip remove=yes dest=/tmp/test.zip"
192.168.1.4 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "archived": [
        "/tmp/test",
        "/tmp/test1"
    ],
    ...
}
192.168.1.4 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
    ...
}
```

Common Function Modules of Ansible – setup and Usage Cases

- The setup module is used to collect information about controlled nodes.
- Requirement: Collect information about the host whose IP address is 192.168.1.4.

```
[root@Ansible ~]# ansible 192.168.1.4 -m setup  
192.168.1.4 | SUCCESS => {  
    "ansible_facts": {  
        "ansible_all_ipv4_addresses": [  
            "192.168.1.4"  
        ],  
        "ansible_all_ipv6_addresses": [],  
        "ansible_apparmor": {  
            "status": "disabled"  
        },  
        "ansible_architecture": "x86_64",  
        "ansible_bios_date": "12/12/2018",  
        "ansible_bios_version": "6.00",  
        "ansible_cmdline": {  
            "BOOT_IMAGE": "/vmlinuz-5.10.0-60.18.  
.....
```

Common Ansible Function Modules – lineinfile and replace

- The lineinfile and replace modules are used to modify file content, which is similar to the function of **sed**.
- Common options for the lineinfile module are as follows:
 - **path**: specifies the absolute path of the file to be modified.
 - **line**: specifies the content to be replaced or inserted
 - **regexp**: uses the regular expression to match content.
 - **insertbefore** or **insertafter**: inserts content before or after the matched line.
- Common options for the replace module are as follows:
 - **path**: specifies the absolute path of the file to be modified.
 - **regexp**: uses the regular expression to match content.
 - **replace**: replaces the content that matches the regular expression.
 - **before** or **after**: modifies the content before or after the content matched by the regular expression.
 - **owner**, **group**, or **mode**: modifies the owner, group, or permission of the file.

Common Ansible Function Modules – user and group

- The user module is used for user management. Common options for the user module are as follows:
 - **name**: specifies the user name.
 - **group** and **groups**: The **group** option is used to set the primary group of a user, and the **groups** option is used to add a user to other groups.
 - **password**: sets the user password.
 - **shell**: sets the user shell.
 - **home**: sets the home directory of the user.
 - **comment**: sets the user description.
 - **uid**: sets the user ID.
- The group module is used for group management. Common options for the group module are as follows:
 - **gid**: indicates the group ID that the user sets.
 - **name**: specifies the group name.

Common Function Modules of Ansible – hostname, cron, yum, and service

- The hostname module is used to change the host name of the controlled host, where the **name** option is used to specify the host name.
- The cron module is used to set scheduled tasks. Common options for the cron module are as follows:
 - **minute, hour, day, month, or weekday:** corresponds to minute, hour, date, month, or weekday in the crontab file respectively.
 - **job:** specifies the task to be executed, which can be a shell command or a script.
 - **name:** sets the task name.
 - **state:** sets the task status. When this option is set to **absent**, it can be used to delete a task.
 - **disable:** enables or disables a task. The **yes** value of this option indicates that the task is disabled, and the **no** value indicates that the task is enabled.
- The yum module is used to manage software packages. Common options for the yum module are as follows:
 - **name:** specifies the name of the software package to be installed.
 - **state:** If this option is set to **absent**, the software is uninstalled. If set to **present**, the software is installed. If set to **lates**, the software of the latest version is installed.
- The service module is used for service management. Common options for the service module are as follows:
 - **name:** specifies the service name.
 - **enabled:** sets the service to start upon system startup.
 - **state:** sets the service status. The **started** value indicates that the service is started. The **stopped** value indicates that the service is stopped. The **reload** value indicates that the service is reloaded.

Contents

1. Introduction to Automated O&M

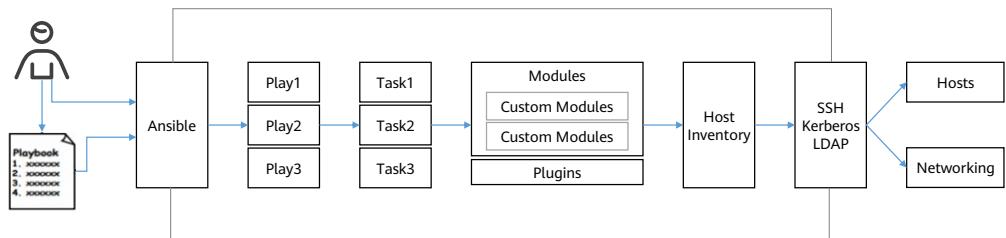
2. Introduction to Ansible

- Ansible Overview and Basic Operations
- Common Ansible Modules
- **Playbook Overview**

3. Introduction to SaltStack

Playbook Overview

- A playbook consists of multiple plays. Each play corresponds to a task. The task invokes the Ansible function module to perform predefined actions based on the preset mechanism.
- Playbooks are compiled using Yet Another Markup Language (YAML).



YAML Features

- YAML has high readability.
- YAML interacts well with script languages.
- YAML is easy to use.
- YAML can be processed based on streams.
- YAML has strong scalability.
- YAML has strict requirements on syntax.

YAML Syntax

- Indentations indicate different levels. The data aligned on the left indicates the same level.
- Indentation must use spaces, instead of tabs.
- If the item starts with a dash (-), it indicates a list and needs to be executed in sequence.
- If the item does not start with a dash (-), it indicates a dictionary and has no execution sequence.

- YAML is case-sensitive.

Core Elements of a Playbook

- **hosts:** specifies the list of controlled hosts.
- **tasks:** specifies the task to be executed.
- **vars:** indicates the variable invoked in the playbook, which can be built-in variables or customized variables.
- **templates:** indicates the template, which is nested internally.
- **handlers** and **notify:** used together to configure triggering conditions for tasks.
- **tags:** indicates the tag, which is used to selectively run some code of a task.

Playbook Commands

- The **ansible-playbook** command is used to run the playbook. The common options are as follows:
 - **--list-hosts**: lists related hosts.
 - **--list-tags**: lists all tags.
 - **--list-tasks**: lists all tasks.
 - **--syntax-check**: only verifies syntax and does not execute the playbook.
 - **-C, --check**: only detects possible changes and does not execute the playbook. The command is usually used to check whether the playbook can run properly.
 - **--start-at-task START_AT_TASK**: starts from a specified task.
 - **-t**: specifies the task corresponding to the specific tags.
 - **--limit**: excludes hosts that do not need to run playbooks.
 - **-v, -vv, or -vvv**: views details.

Basic Usage of Playbooks – hosts and tasks

- Requirement: Install zabbix-agent2 in the Nginx host group, set the zabbix-server address to **192.168.1.4**, and ensure that zabbix-agent2 automatically starts upon system startup.

```
---
- hosts: Nginx
  remote_user: root
  gather_facts: no

  tasks:
    - name: Download and install the Zabbix Yum repository
      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
      yum:
        name=zabbix-agent2
        state=present
    - name: Modify the zabbix-agent2 configuration file and set the server address to 192.168.1.4
      replace:
        path: /etc/zabbix/zabbix_agent2.conf
        regexp: '^Server=127.0.0.1$'
        replace: 'Server=192.168.1.4'
    - name: Set zabbix-agent2 to start upon system startup
      service:
        name: zabbix-agent2
        state: started
        enabled: yes
```

- Generally, a YAML file is named in the format of YAML or YML. The first line at the beginning is three hyphens (---). You are advised to leave a blank line between **hosts** and **tasks** to improve playbook readability.
- hosts** can be represented by logical relationships, for example, **test1:test2** and **test1::&test2**.
- remote_user** can be used in **hosts** or **tasks**, indicating the user who performs the operation.
- gather_facts** collects host information. The function of **gather_facts** is the same as that of the setup module. Set to **yes** to collect variables about remote hosts, and **no** otherwise. If you do not need to collect host information, you are advised to set **gather_facts** to **no** to improve playbook efficiency.

Basic Usage of Playbooks – handlers and notify

- Requirement: Install zabbix-agent2 in the Nginx host group. zabbix-agent2 automatically restarts each time the zabbix-server address is changed.

```
---
- hosts: Nginx
  remote_user: root
  gather_facts: no

  tasks:
    .....
    - name: Modify the zabbix-agent2 configuration file and set the server address to 192.168.1.4
      replace:
        path: /etc/zabbix/zabbix_agent2.conf
        regexp: '^Server=127.0.0.1$'
        replace: 'Server=192.168.1.4'
      notify: restart agent2
    - name: Set zabbix-agent2 to start upon system startup
      service:
        name: zabbix-agent2
        state: started
        enabled: yes

  handlers:
    - name: restart agent2
      service:
        name: zabbix-agent2
        state: restarted
```

- Multiple **notify** messages can be set in a task to trigger multiple **handlers**.

Basic Usage of Playbooks – tags

- Requirement: Install zabbix-agent2 in the Nginx host group. Define a tag for the action of changing the zabbix-server address. If the zabbix-server address changes, you only need to perform this operation.

```
---
- hosts: Nginx
  remote_user: root
  gather_facts: no

  tasks:
  .....
    - name: Modify the zabbix-agent2 configuration file and set the server address to 192.168.1.4
      tags: modify-server-IP
      replace:
        path: /etc/zabbix/zabbix_agent2.conf
        regexp: 'Server=127.0.0.1$'
        replace: 'Server=192.168.1.4'
        notify: restart agent2
    - name: Set zabbix-agent2 to start upon system startup
      service:
        name: zabbix-agent2
        state: started
        enabled: yes
  .....
```

- If you only need to change the IP address, run the command: **ansible-playbook -t modify-server-IP zabbix-agent2.yml**.

Basic Usage of Playbooks – vars

- Requirement: Create a user and group Nginx in the Nginx host group.

```
##### Implementation method 1: Define variables in a playbook and
set values when running commands.
---
- hosts: Nginx
  remote_user: root
  gather_facts: no

  tasks:
    - name: Create a group
      group:
        name={{ group_name }}
    - name: Create a user
      user:
        name={{ user_name }}
```

Run the following command:
ansible-playbook -e group_name=nginx -e user_name=nginx var.yaml

```
### Implementation method 2:
Define variables and assign
values to the variables in a
playbook.
---
- hosts: Nginx
  remote_user: root
  gather_facts: no
  vars:
    - group_name: nginx
    - user_name: nginx

  tasks:
    - name: Create a group
      group:
        name={{ group_name }}
    - name: Create a user
      user:
        name={{ user_name }}
```

```
### Implementation method 3: Create a
variable file separately and invoke it in the
playbook.
### The content of the variable file vars.yml
is as follows:
---
group_name: nginx
user_name: Nginx
### The content of the playbook is as
follows:
---
- hosts: Nginx
  remote_user: root
  gather_facts: no
  vars_files:
    - vars.yml

  tasks:
    - name: Create a group
      group:
        name={{ group_name }}
    - name: Create a user
      user:
        name={{ user_name }}
```

- vars** is the variable in Ansible. Use **{{variable}}** to define variables or use variables defined in setup, and specify values in commands, playbooks, or files.

Basic Usage of Playbooks – vars

- Requirement: Change the host name in the Nginx host group to the Nginx-*num* format.

```
### Modify the host group configuration file as follows:  
[Nginx]  
192.168.1.14 host=01  
192.168.1.15 host=02  
[Nginxvars]  
group=Nginx  
  
### Use the hostname module to change host names in a unified manner.  
ansible Nginx -m hostname -a "name={{ group }}-{{ host }}"
```

Basic Usage of Playbooks – templates

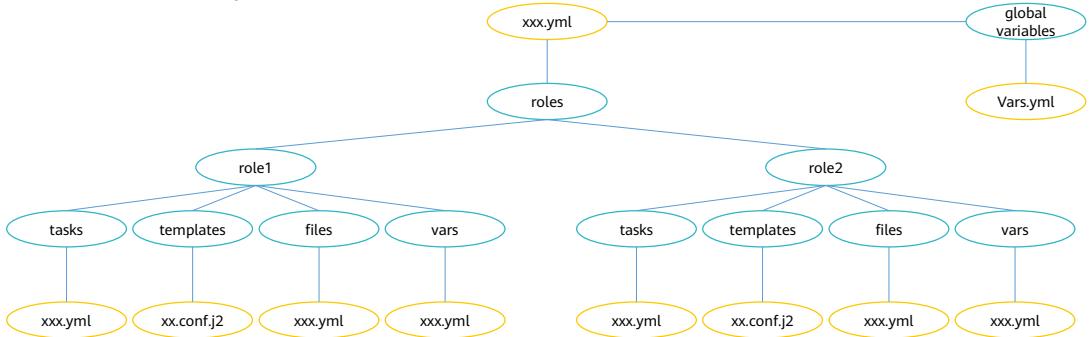
- **templates** can dynamically generate configuration files for playbooks to invoke by referring to Ansible module files. Templates files must be stored in a fixed directory and end with J2.
- Requirement: Install zabbix-agent2 on the hosts in the Nginx host group and use **zabbix_agent2.conf.j2** as the configuration file.

```
---
- hosts: Nginx
  remote_user: root
  gather_facts: no

  tasks:
    - name: Download and install the Zabbix Yum repository
      .....
    - name: Modify the zabbix-agent2 configuration file by using the same template
      template: src=/root/template/zabbix_agent2.conf.j2 dest=/etc/zabbix/zabbix_agent2.conf
    - name: Set zabbix-agent2 to start upon system startup
      service:
        name: zabbix-agent2
        state: started
        enabled: yes
```

Basic Usage of Playbooks – roles

- **roles** is used for hierarchical and structured playbooks. Variable files, tasks, and handlers can be automatically loaded based on defined and hierarchical directories to complete complex O&M tasks in batches.
- The common directory structure of **roles** is as follows:



Basic Usage of Playbooks – roles

- Requirements: Deploy WordPress in a cluster. The database is based on MySQL, and the HTTP server is implemented using Apache.

```
---
```

```
- name: WordPress
  remote_user: root
  hosts: all
  roles:
    - apache
    - mysql
    - php
    - wordpress
```

Contents

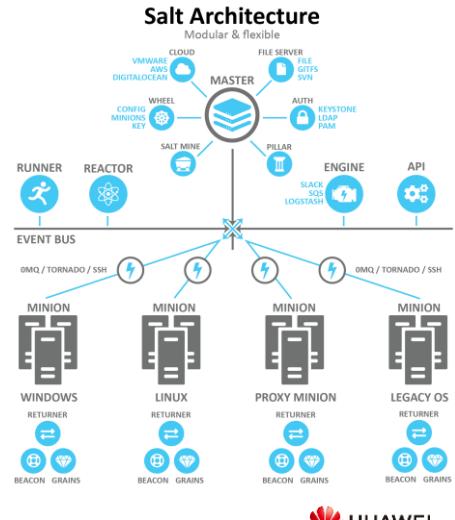
1. Introduction to Automated O&M
2. Introduction to Ansible
- 3. Introduction to SaltStack**

Introduction to SaltStack

- Based on the Salt project, SaltStack consists of two function modules:
 - Configuration management system
 - Distributed remote command execution system
- SaltStack has six features:
 - Usability
 - Parallel execution
 - Maturity and stability
 - Easy integration
 - High performance
 - Open source

Logical Architecture of SaltStack

- Salt master and salt minion
 - SaltStack uses a client-server architecture, where the host server is the salt master and the client is the salt minion. The master sends commands to the minion, and the minion registers with a specific master.
 - SaltStack also uses publish-subscribe pattern, where the master is the publisher of commands, and the minion obtains said commands.
- Targets
 - The master defines a group of minions that should execute jobs as targets.
- Open event system
 - The event bus is used for internal communication between the master and minion in the event management system.
 - ZeroMQ and Tornado are pluggable components used by the minion to subscribe to jobs and execution results from the event management system.
- Grains and pillar
 - Grains run in the minion and are used to collect information about the minion, such as the OS, domain name, and IP address. Grains are static data and are updated only when the system information changes or a parameter is customized.
 - Pillar stores dynamic data defined by the master and pushes the data to the minion when necessary.
- Beacons and reactors
 - Beacons and reactors are used for monitoring. Beacons run in the minion. In the event of a fault, beacons trigger reactors to provide help information for troubleshooting.



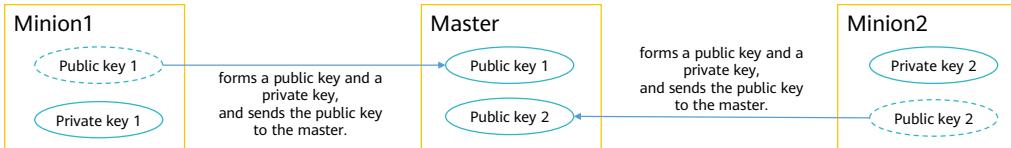
50 Huawei Confidential



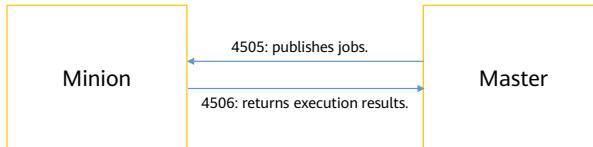
- Architecture diagram source:
https://docs.saltproject.io/en/latest/topics/salt_system_architecture.html#salt-masters-and-salt-minions
- Pillars and grains are used to store data. Pillar is used to store private data, such as user names and passwords, which can be viewed only by specific minions.
- In addition to running the salt-minion service, the master can distribute commands to the client through the proxy and SSH. The salt-minion service is not supported by the proxy.
- Run the **salt * grains.ls** command to query the grains item of all hosts, and the **salt * grains.items** command to display more details.

Authentication and Communication Between the Master and Minion

- The master and minion use public and private keys for authentication.



- The master communicates with the minion through 4505 and 4506.



- When the minion is started for the first time, the private key **minion.key** and public key **minion.pub** are automatically generated in **/etc/salt/pki/minion**, and the public key is sent to the master.
- When the minion sends a public key to the master, the master does not automatically receive the public key. The administrator needs to use **salt-key** to manually receive the public key and report the error to **/etc/salt/pki/masker/minions**.

SaltStack Installation and Deployment

- Check the installation environment.
- Install salt-master and salt-minion.
 - Bootstrap installation: uses the script provided by SaltStack.
 - Manual installation: uses the Yum repository.
- Configure SaltStack after the installation.
 - Configure the master to be registered in the minion.
- Start the service.
- Configure the authentication between the master and minion.
- Check the configuration.

Important Configurations of the Master and Minion

- The default configuration file of the master is **/etc/salt/master.d**, which contains all CONF files in **/etc/salt/master.d**.

```
# The address of the interface to bind to:  
interface: 0.0.0.0  
# The tcp port used by the publisher:  
publish_port: 4505  
# The port used by the communication interface. The ret (return) port is the  
# interface used for the file server, authentication, job returns, etc.  
ret_port: 4506  
# Directory used to store public key data:  
pki_dir: /etc/salt/pki/master  
file_roots:  
  base:  
    - /srv/salt/  
pillar_roots:  
  base:  
    - /srv/pillar
```

- The default configuration file about the master in the minion is **/etc/salt/master**, which contains all CONF files in **/etc/salt/master.d**.
- The default configuration file about the minion in the minion is **/etc/salt/minion**, which contains all CONF files in **/etc/salt/master.d**.

```
# Set the location of the salt master server. If the master server cannot be  
# resolved, then the minion will fail to start.  
master: salt  
id:
```

- The directory set by **file_roots** is used to store job execution scripts.
- pillar_roots** is used to set the directory for storing pillar data.

Mutual-trust Authentication Configuration Between the Master and Minion

- Run the **salt-key** command on the master to receive the key sent by the minion. The common options are as follows:
 - **-A**: receives all keys.
 - **-D**: deletes all keys.
 - **-L**: lists all keys.
 - **-a**: receives the specified key.
 - **-j**: rejects the specified key.
 - **-d**: deletes the specified key.

Basic Operations of SaltStack – Grains

- Grains are used to store static data of the minion. Administrators can define targets through built-in grains items or customized items.

- Run the **salt '*' grains.items** command to view all built-in items of the system.

```
[root@SaltStack ~]# salt '*' grains.items
Nginx1:
-----
biosreleasedate:
    12/12/2018
.....
```

- Run the **salt '*' grains.item item_name** command to view the value of a specified item.

```
[root@SaltStack ~]# salt '*' grains.item os
Nginx2:
-----
os:
    openEuler
Nginx1:
-----
os:
    openEuler
```

- Run the **salt -G 'item_name:item_value' test.ping** command to specify a host, based on the item value, to run the command.

```
[root@SaltStack ~]# salt -G 'ipv4:192.168.1.14' test.ping
Nginx1:
    True
```

- The asterisk (*) in the command indicates all hosts. You can specify a host to view the information.

Basic Operations of SaltStack – Grains

- Modify the grains part in the minion configuration file to customize the item.
 - Create a grains configuration file in the **/etc/salt/minion.d** directory on the minion node and enter the following content:

```
grains:  
  role: Nginx
```
 - After the salt-minion service is restarted, check the customized item on the master node.

```
[root@SaltStack ~]# salt '*' grains.item role  
Nginx2:  
-----  
  role:  
Nginx1:  
-----  
  role:  
    Nginx
```

- In addition to customizing grains item in the **/etc/salt/minion.d** directory, you can create grains files or customize the item in the **/etc/salt** directory. In grains, the item is defined in the **item: value** format. SaltStack can identify related content.
- After grains are modified, the master needs to synchronize the modification in either of the following ways:
 - Restart the salt-minion service.
 - Run the **salt '*' saltutil.sync_grains** command on the master.

Basic Operations of SaltStack – Pillar

- Pillar is used to store dynamic data and is valid only for specified minions.
- Pillar is defined by creating a configuration file in a specified directory. For example, create a pillar configuration file in the **/srv/pillar** directory and enter the following content:

```
[root@SaltStack pillar]# cat test.sls
test: Nginx1
[root@SaltStack pillar]# cat top.sls
base:
  'Nginx1':
    - test
```

- After the configuration is complete, run the following command to check whether the customized pillar takes effect:

```
[root@SaltStack pillar]# salt '*' saltutil.refresh_pillar
...
[root@SaltStack pillar]# salt '*' pillar.items test
Nginx2:
-----
  test:
Nginx1:
-----
  test:
  Nginx1
```

- The directory that stores the pillar configuration file is specified in the master configuration file. In this course, the storage directory is in the **/srv/pillar** directory.

SaltStack Remote Execution Function

- The command format of SaltStack remote execution is **salt [options] '<target>' <function> [arguments]**.
 - options:** The mode of setting the target
 - target:** The client on which the command is to be remotely executed
 - function:** The function module
 - arguments:** The command to be executed

```
[root@SaltStack ~]# salt -L 'Nginx1,Nginx2' cmd.run 'ip addr'
Nginx1:
.....
2: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:0c:29:88:a6:a9 brd ff:ff:ff:ff:ff:ff
    altname enp11s0
    inet 192.168.1.14/24 brd 192.168.1.255 scope global noprefixroute ens192
        valid_lft forever preferred_lft forever
.....
Nginx2:
.....
2: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:0c:29:a8:db:82 brd ff:ff:ff:ff:ff:ff
    altname enp11s0
    inet 192.168.1.15/24 brd 192.168.1.255 scope global noprefixroute ens192
        valid_lft forever preferred_lft forever
.....
```

SaltStack Remote Execution Function – Specifying the Target

- SaltStack supports multiple methods of specifying a target. The common methods are as follows:
 - Wildcard mode, for example, `salt '*' test.ping`, `salt'Nginx*' test.ping`, `salt'Nginx[1|2]' test.ping`, and `salt'Nginx[!1]' test.ping`.
 - `-L`: indicates the list mode, for example, `salt -L 'Nginx1,Nginx2' test.ping`.
 - `-E`: indicates the regular expression mode, for example, `salt -E '^N' test.ping`.
 - `-I`: specifies the host using pillar, for example, `salt -I 'test:Nginx1' test.ping`.
 - `-G`: specifies the host using grains, for example, `salt -G 'ipv4:192.168.1.14' test.ping`.
 - `-C`: specifies the host in hybrid mode, for example, `salt -C 'I@test:Nginx1 or G@ipv4:192.168.1.15' test.ping`.
 - `-S`: specifies the host using network information, for example, `salt -S '192.168.1.0/24' test.ping`, and `salt -S '192.168.1.15' test.ping`.
 - `-H`: specifies the host in a host group. Set the host group in the configuration file first, for example, `salt -N test test.ping`.

SaltStack Remote Execution Function – Function Modules

- The function module of SaltStack is developed using Python and is used to execute jobs in batches. For details, see the following official document:
 - <https://docs.saltproject.io/en/latest/ref/modules/all/index.html#all-salt-modules>
- Query the module installed on the minion: **salt 'minion_name' sys.list_modules**
- Query the module usage document: **salt 'minion_name' sys.doc 'module_name'**
- Query the function supported by the module: **salt 'minion_name' sys.list_functions 'module_name'**
- Common SaltStack modules:
 - pkg
 - network
 - service
 - status
 - file
 - user and group

Common Function Modules of SaltStack – pkg

- The `pkg` module is used for package management. The common functions are as follows:
 - `pkg.available_version`: checks the latest version of the software package.
 - `pkg.download`: downloads the installation package but does not install it. After the download is complete, the directory where the installation package is stored is returned.
 - `pkg.list_repos`: lists information about the Yum repository.
 - `pkg.install`: installs the specified software package. `pkg.group_install` installs the specified software group.
 - `pkg.remove`: uninstalls the specified software package.
 - `pkg.owner`: checks which software package a file belongs to.
 - `pkg.list_pkgs`: lists all installed software packages.
 - `pkg.mod_repo`: modifies some parameters of the Yum repository.

Common Function Modules of SaltStack – network

- The network module is used to collect and manage network information. The common functions are as follows:
 - **network.active_tcp**: returns active TCP connections in dictionary format.
 - **network.connect**: tests the connectivity with a host through a specified port of the minion.
 - **network.default_route**: checks the default route of the minion.
 - **network.get_route**: checks route details of a specified network segment of the minion.
 - **network.dig**: uses `dig` to perform a DNS test on the minion.
 - **network.interface**: checks the address of the port specified by the minion.
 - **network.ping**: specifies a minion to perform a ping test on an address.
 - **network.routes**: checks the routing table of the specified minion.
 - **networktraceroute**: specifies the minion to trace the route of an address.

Common Function Modules of SaltStack – service

- The service module is used for service management. The common functions are as follows:
 - **service.available**: checks whether a service is available. If yes, **True** is returned. Otherwise, **False** is returned.
 - **service.disable**: disables the service from automatically starting upon system startup.
 - **service.disabled**: checks the service is listed to not automatically start upon system startup. If listed, **False** is returned; if not listed, **True** is returned.
 - **service.enable**: enables the service to automatically start upon system startup.
 - **service.enabled**: checks whether the service is in the list of the services that automatically start upon system startup. If yes, **True** is returned. Otherwise, **False** is returned.
 - **service.get_all**: lists all installed and available services.
 - **service.start**: starts a service.
 - **service.stop**: stops a service.
 - **service.restart**: restarts a service.
 - **service.reload**: reloads the configuration of a service.

Common Function Modules of SaltStack – file

- The `file` module is used to manage regular and non-regular files and directories, such as setting users, groups, and permissions for files. The common functions are as follows:
 - `file.access`: tests whether SaltStack has a permission on the file or directory. `f` checks whether the queried object exists. `r`, `w`, and `x` are used to check whether SaltStack has the read, write, and execute permissions.
 - `file.append`: adds content to a file. If the file does not exist, the system reports an error.
 - `file.basename` or `file.dirname`: specifies a file by a path and SaltStack returns the last part or the directory, respectively.
 - `file.touch`: creates a file. If the file already exists, the access time or modification time of the file is updated.
 - `file.chgrp` or `file.chown`: changes the group or the user and group to which a file or directory belongs, respectively.
 - `file.copy`: copies the file.
 - `file.find`: searches for the file.
 - `file.diskusage`: calculates the space occupied by the directory. The unit of the returned value is byte.
 - `file.get_gid`, `file.get_group`, `file.get_uid`, and `file.get_user`: obtains the ID of the group that owns the file, the group that owns the file, the ID of the user that owns the file, or the user that owns the file.
 - `file.grep`: searches for specified content in a file.
 - `file.makedirs` or `file.mkdir`: creates directories. The two functions have certain differences. For example, when you create the `/tmp/test` directory, if `file.makedirs` is used, enter `/tmp/test/` after it, and if `file.mkdir` is used, enter `/tmp/test` instead. In addition, `file.makedirs` does not support creating multiple directories at the same time, but `file.mkdir` supports.
 - `file.move`: moves a file or directory.
 - `file.remove` or `file.rename`: deletes or renames files.
 - `file.rmdir`: deletes the empty directory.

Common Function Modules of SaltStack – status

- The status module is used to view the minion status. The common functions are as follows:
 - **status.cpuinfo**: displays the minion CPU information.
 - **status.cpustats**: checks the minion CPU status.
 - **status.diskstats**: checks the minion drive status.
 - **status.diskusage**: checks the usage of a drive on the minion.
 - **status.meminfo**: checks the minion memory information.
 - **status.netstats**: checks the minion network status.

SaltStack Configuration Management Function

- The SaltStack configuration management depends on the **state** function module.
- The SaltStack configuration file uses the YAML syntax (the file name extension must be SLS) and must be stored in the specified directory on the master node. The directory is specified by the **file_roots** parameter.
- With custom multiple configuration files, the **top.sls** file can be used as the entry of all configurations and files to automatically run all SLS files.

- The top file is called **top.sls** by default and can be changed in the master configuration file.
- The storage location of the **top.sls** file is specified by **base** in the **file_roots** parameter.
- To make a clear directory structure, the **top.sls** file is stored in the top-level directory in the **base** directory, and other SLS files are stored in the second-level directory.

SaltStack Configuration Management Function – Configuration File Compilation

- Requirement: Use the SaltStack configuration management function to install the NGINX service for Nginx1 and set the service to automatically start upon system startup.

```
[root@SaltStack nginx]# pwd  
/srv/salt/nginx  
[root@SaltStack nginx]# cat nginx.sls  
---  
nginx_install:  
pkg.installed:  
    - name: nginx  
enable_nginx:  
service.running:  
    - name: nginx  
    - enable: True  
[root@SaltStack nginx]# salt 'Nginx1' state.sls nginx
```

SaltStack Configuration Management Function – Top File Compilation

- Requirement: Based on the preceding slide, install the httpd service on Nginx2, set the service to start upon system startup, and use the top file to execute two SLS configuration files.

```
[root@SaltStack salt]# pwd  
/srv/salt  
[root@SaltStack salt]# ls  
httpd.sls Nginx.sls top.sls  
[root@SaltStack salt]# cat top.sls  
base:  
'Nginx1':  
    - nginx  
'Nginx2':  
    - httpd
```

```
[root@SaltStack httpd]# pwd  
/srv/salt/httpd  
[root@SaltStack httpd]# cat httpd.sls  
---  
httpd_install:  
pkg.installed:  
    - name: httpd  
enable_httpd:  
service.running:  
    - name: httpd  
    - enable: True
```

- Before using the top file, delete the comment of the top file configuration in the configuration file.
- After the top file is compiled, run the **salt '*' state.highstate** command to execute the tasks in the top file for all minions.
- Before running the top file, you are advised to run the **salt '*' state.highstate test=True** command to check whether the top file can be correctly executed.

Quiz

1. (Single-answer) Which of the following Ansible function modules is used to send files from the control host to the controlled host?
 - A. File
 - B. Copy
 - C. Fetch
 - D. Move

- B

Summary

- This chapter describes the basic usage of Ansible and SaltStack, including basic function modules and compilation of automatic job execution scripts.

Acronyms and Abbreviations

| Acronym/Abbreviation | Full Spelling |
|----------------------|-----------------------------------|
| API | Application Programming Interface |
| SSH | Secure Shell Protocol |
| YAML | Yet Another Markup Language |

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and
organization for a fully connected,
intelligent world.

Copyright©2024 Huawei Technologies Co., Ltd.
All Rights Reserved.

The information in this document may contain predictive
statements including, without limitation, statements regarding
the future financial and operating results, future product
portfolio, new technology, etc. There are a number of factors that
could cause actual results and developments to differ materially
from those expressed or implied in the predictive statements.
Therefore, such information is provided for reference purpose
only and constitutes neither an offer nor an acceptance. Huawei
may change the information at any time without notice.



Advanced Shell Scripts



Foreword

- This course describes the advanced functions of shell scripts, including conditional judgment, functions, regular expressions, and grep, sed, and AWK commands.

Objectives

- Upon completion of this course, you will understand:
 - Conditional judgment.
 - Functions and common operations.
 - grep syntax and common operations.
 - Regular expressions.
 - sed syntax and common operations.
 - AWK syntax and common operations.

Contents

- 1. Conditional Judgment in Shell Scripts**
2. Functions in Shell Scripts
3. Introduction to grep Commands
4. Introduction to Regular Expressions
5. Introduction to sed
6. Introduction to AWK Statements

Common Conditional Judgment Methods in Shell Scripts

- Conditional judgment controls the subsequent operations of scripts by checking whether specified conditions are met.
- Common conditional judgement methods:
 - test
 - Brackets ([])
 - Double brackets ([[]])
 - Double parentheses ((()))

Introduction to test

- Syntax:
 - The **test** expression: **test 1 -eq 1**
- If the condition listed in the **test** command is met, the **test** command exits and returns the exit status code **0**. If the condition is not met, the **test** command exits and returns a non-zero exit status code, which prevents the if-then statement from being executed.

```
[root@openEuler ~]# if test 1 -eq 1;then echo "a=b";fi  
a=b
```

Application Scenarios of test – Numeric Comparison

```
[root@openEuler ~]# cat num.sh
#!/bin/bash

a=$1
b=$2
if test $a -eq $b
then
    echo "a=b"
fi
if test $a -ne $b
then
    echo "a!=b"
fi
if test $a -gt $b
then
    echo "a>b"
fi
if test $a -lt $b
then
    echo "a<b"
fi
```

```
if test $a -ge $b
then
    echo "a>=b"
fi
if test $a -le $b
then
    echo "a<=b"
fi
[root@openEuler ~]# sh num.sh 1 2
a=b
a<b
a<=b
[root@openEuler ~]#
```

| Option | Description |
|--------|--------------------------|
| -eq | Equal to |
| -ne | Not equal to |
| -gt | Greater than |
| -lt | Less than |
| -ge | Greater than or equal to |
| -le | Less than or equal to |

- The **test** command cannot be used to compare floating-point numbers.
- When the double parentheses ((())) method is used, compare numbers by using =, >, <, >=, and <=.

Application Scenarios of test – Character String Comparison

```
[root@openEuler ~]# cat string.sh
str1=$1
str2=$2
if test -n "$str1"
then
    if test -n "$str2"
    then
        if test "$str1" = "$str2"
        then
            echo "str1=str2"
        fi
        if test "$str1" != "$str2"
        then
            echo "str1!=str2"
        fi
    elif test -z "$str2"
    then
        echo "str2 is null"
    fi
fi
```

```
elif test -z "$str1"
then
    echo "str1 is null"
fi
[root@openEuler ~]# sh string.sh aa bb
str1=aa
[root@openEuler ~]# sh string.sh aa
str2 is null
[root@openEuler ~]# sh string.sh
str1 is null
[root@openEuler ~]#
```

| Option | Description |
|--------|------------------|
| -n | Non-empty string |
| -z | Empty string |
| = | Equal to |
| != | Not equal to |

- When comparing with character strings, use an escape character (\) before the less-than sign (<) and the greater-than sign (>), for example, **if test "\$str1" \> "\$str2"**.

Application Scenarios of test – File Comparison

```
[root@openEuler ~]# mkdir my_dir
[root@openEuler ~]# chmod +x num.sh
[root@openEuler ~]# ll
total 40K
drwxr-xr-x 2 root root 4.0K Mar 29 07:15 my_dir
-rwxr-xr-x 1 root root 276 Mar 29 06:30 num.sh
-rw-r--r-- 1 root root 313 Mar 29 07:06 string.sh
[root@openEuler ~]#
[root@openEuler ~]# if test -d my_dir;then echo "It is a directory.";fi
It is a directory.
[root@openEuler ~]# if test -f string.sh;then echo "It is a file.";fi
It is a file.
[root@openEuler ~]# if test -e string.sh;then echo "This file exists.";fi
This file exists.
[root@openEuler ~]# if test -r string.sh;then echo "This file can be read.";fi
This file can be read.
[root@openEuler ~]# if test -w string.sh;then echo "This file can be written.";fi
This file can be written.
[root@openEuler ~]# if test -x num.sh;then echo "This file can be executed.";fi
This file can be executed.
```

| Option | Description |
|--------|----------------------------------|
| -d | Directory |
| -f | Common file |
| -e | Whether the file exists |
| -r | Whether the file can be read |
| -w | Whether the file can be written |
| -x | Whether the file can be executed |

- In the old and new file comparison, old indicates that the file is created earlier.
 - **file1 -nt file2:** checks whether file1 is created later than file2.
 - **file1 -ot file2:** checks whether file1 is created earlier than file2.
- **-s:** checks whether there is a file and is non-empty or not.

Application Scenarios of test – Logical Judgment

- Use **-a** or **-o** for the logical judgment. Specifically,
 - test *expression 1 -a expression 2*
 - test *expression 1 -o expression 2*

```
[root@openEuler ~]# if test 1 -eq 1 -a 2 -eq 2;then echo "ok";fi
ok
[root@openEuler ~]#
[root@openEuler ~]#
[root@openEuler ~]# if test ! 1 -eq 1 -a 2 -eq 2;then echo "ok";fi
[root@openEuler ~]#
[root@openEuler ~]#
[root@openEuler ~]# if test 1 -eq 2 -o 2 -eq 2;then echo "ok";fi
ok
[root@openEuler ~]#
```

| Option | Description |
|-----------|---|
| -a | Checks whether the two conditions are met at the same time. |
| -o | Checks whether at least one of the two conditions is met. |
| ! | Indicates the negation operation. |

How to Use []

- The [] command has the same functions, standard numeric comparison and character string comparison, as those of the **test** command.
- Syntax:
 - [*expression*]
- There must be a space before and after *expression*.

How to Use [[]]

- The [[]] command provides advanced features for character string comparison.
- Syntax:
 - [[*expression*]]
- There must be a space before and after *expression*.
- The *expression* in this command uses the standard character string comparison used in the **test** command. It also provides another feature, pattern matching, which is not provided by the **test** command.

How to Use (())

- Advanced mathematical expressions can be used in the (()) command for comparison.
- Syntax:
 - ((*expression*))
- The *expression* can be any mathematical assignment or comparison expression. In addition to the standard mathematical operators used by the **test** command, the table on the right lists other operators used in the (()) command.

| Operator | Function |
|--------------|----------------|
| <VARIABLE>++ | Post-increment |
| <VARIABLE>-- | Post-decrement |
| ++<VARIABLE> | Pre-increment |
| --<VARIABLE> | Pre-decrement |
| ! | Logical NOT |
| ~ | Bitwise NOT |
| ** | Exponentiation |
| << | Left shift |
| >> | Right shift |
| & | Bitwise AND |
| | Bitwise OR |
| && | Logical AND |
| | Logical OR |

- The syntax of the (()) command is similar to that of C.

Introduction to the Logical Judgement of [], [[]], and (())

- [], [[]], and (()) can be used for logical judgment. Single-statement judgment is used to control execution.
 - **a && b** : If **a** is true, then execute **b**.
 - **a || b** : If **a** is false then execute **b**.
 - When multiple **&&** or **||** are used together, the judgment is as follows:
 - If the command before **&&** is executed successfully, the next command is executed.
 - If the command before **||** is executed successfully, the next command is not executed.

| Option | Description |
|--------|-------------|
| && | Logical AND |
| | Logical OR |
| ! | Logic NOT |

```
[root@openEuler ~]# [ 1 -eq 1 ] && echo a
a
[root@openEuler ~]# [ 1 -eq 2 ] || echo a
a
[root@openEuler ~]# [ 1 -eq 2 ] && echo a || echo b || echo c
b
[root@openEuler ~]# [ 1 -eq 2 ] || echo a && echo b && echo c
a
b
c
[root@openEuler ~]# ! [ 1 -eq 2 ] && echo a
a
```

- If the command is executed successfully, the value is **true**. For example, **echo a** is **true**.

Summary of Conditional Judgment Syntax

- Conditional judgment is used in process control statements.
- Syntax:
 - The **test** expression: **test 1 -eq 1**
 - The brackets command: [*expression*]. There must be a space before and after the *expression*, for example, [1 -eq 1].
 - The double brackets command: [[*expression*]]. It supports more features than the single bracket command does. There must be a space before and after *expression*, for example, [[1 -eq 1]].
 - The double parentheses command: ((*expression*)), which is similar to the syntax of C. No space before and after the expression, for example, ((1==1)).

```
[root@openEuler ~]# if test 1 -eq 1;then echo "a=b";fi  
a=b  
[root@openEuler ~]# if [ 1 -eq 1 ];then echo "a=b";fi  
a=b  
[root@openEuler ~]# if [[ 1 -eq 1 ]];then echo "a=b";fi  
a=b  
[root@openEuler ~]# if ((1==1));then echo "a=b";fi  
a=b
```

- There are different structures of syntax. Choose one according to your needs and personal habits.

Conditional Judgment Used for the for Loop Nesting.

```
[root@openEuler ~]# cat for.sh
for i in 1 2 3
do
    for j in A B C
    do
        if [ $i -eq 1 ]
        then
            echo "$j is one year old now."
        elif [ $i -eq 2 ]
        then
            echo "$j is two years old now."
        elif [ $i -eq 3 ]
        then
            echo "$j is three years old now."
        else
            echo "I don't know how old he is."
        fi
    done
```

```
if [ $i -lt 3 ]
then
    echo "Happy New Year!"
fi
done
[root@openEuler ~]# sh for.sh
A is one year old now.
B is one year old now.
C is one year old now.
Happy New Year!
A is two years old now.
B is two years old now.
C is two years old now.
Happy New Year!
A is three years old now.
B is three years old now.
C is three years old now.
[root@openEuler ~]#
```

Conditional Judgment Used for the while Loop Nesting.

```
[root@openEuler ~]# cat while.sh
i=1
while [ $i -le 9 ]
do
    j=1
    while [ $j -le $i ]
    do
        echo -n "$i*$j=$[i*j]"
        let j++
    done
    let i++
    echo ""
done
[root@openEuler ~]#
```

```
[root@openEuler ~]# sh while.sh
1*1=1
2*2=4
3*3=9
4*4=16
5*5=25
6*6=36
7*7=49
8*8=64
9*9=81
[root@openEuler ~]#
```

Contents

1. Conditional Judgment in Shell Scripts
- 2. Functions in Shell Scripts**
3. Introduction to grep Commands
4. Introduction to Regular Expressions
5. Introduction to sed
6. Introduction to AWK Statements

Function Structure

- Shell allows a group of commands or statements to form a code block. This code block is called a shell function and can be invoked by using a function name.
- Syntax:
 - The standard definition is **function fun(){}.** There are two simple definitions: **fun(){} and function fun{}.** No parameter is contained in the parentheses and braces.
 - The function must be defined before being invoked.

```
[root@openEuler ~]# cat fun.sh
function hello(){
    echo "Hello"
}
world(){
    echo "Wolrd"
    return 2
}
hello
echo $?
```

```
# Use the source command to execute fun.sh to load
the functions or variables in the script to the current
shell.
[root@openEuler ~]# source fun.sh
Hello
0
[root@openEuler ~]# world
World
[root@openEuler ~]# echo $?
2
[root@openEuler ~]#
```

- The common syntax is the simple definition **func(){}.**

Setting the Function Exit Status Code

- The **return** command is used to exit a function and return a specific exit status code. This code can be defined by a specified integer ranging from 0 to 255.
- The variable **\$?** returns the exit status code of the last executed command.
- Use **\$?** immediately after the function ends to check the return value. If other commands are executed before using **\$?**, the return value of the function is lost.

- Default exit status code:
 - **0:** The command is executed successfully.
 - **1:** Common unknown error occurs.
 - **2:** The command parameter is incorrect.

Viewing and Deleting Functions

- The **set** command can be used to view all variables and functions in the current shell.
- The **unset** command can be used to delete a variable or function in the current shell.
- **Declare -F** can be used to view the function list, and **declare -f** can be used to view the function definition.

```
[root@openEuler ~]# set
.....
whoiam=root
.....
hello ()
{
    echo "Hello"
}
world ()
{
    echo "Wolrd";
    return 2
}
```

```
[root@openEuler ~]# unset world
[root@openEuler ~]# set
.....
whoiam=root
.....
hello ()
{
    echo "Hello"
}
[root@openEuler ~]# world
-bash: world: command not found
[root@openEuler ~]#
```

Reading the Output Value of a Function

- If a function defines the output after the running is complete, run the **echo** command to read the output value.
- For example, assign the output value of the **dbl** function to the variable **result**.

```
[root@openEuler ~]# cat test.sh
#!/bin/bash
function dbl {
    read -p "Enter a value: " value
    echo $[ $value * 2 ]
}

result=$(dbl)
echo "The new value is $result"
[root@openEuler ~]# sh test.sh
Enter a value: 200
The new value is 400
```

- The function uses the **echo** statement to display the calculation result. This script obtains the output of the **dbl** function instead of viewing the exit status code.

Parameter Passing and Recursion of Functions

- The parameter passing of both functions and scripts uses `$n` to indicate the n th parameter.
- The function body can invoke the function itself to implement recursive invoking of the function.

```
# Factorial of numbers
[root@openEuler ~]# cat fun2.sh
myfun(){
    if [ $1 -eq 1 ]
    then
        echo 1
    else
        temp=${$1-1}
        result=$(myfun $temp)
        echo ${$1}*${result}
    fi
}
echo "$($myfun 5)"
[root@openEuler ~]# bash fun2.sh
120
[root@openEuler ~]#
```

Function Library File

- The bash shell allows you to create a function library file and then reference the library file in multiple scripts.

Create a library function file.

```
[root@openEuler ~]# cat myfuncs
# my script functions
function addem
{
    echo $[ $1 + $2 ]
}
```

Run the following library function file:

```
[root@openEuler ~]# cat test.sh
#!/bin/bash
./myfuncs
value1=10
value2=5
result1=$(addem $value1 $value2)
echo "The result of adding them is: $result1"
```

```
[root@openEuler ~]# sh test.sh
The result of adding them is: 15
```

- Use the source command in the `./functionname` format to run the library file script in the shell script.

Contents

1. Conditional Judgment in Shell Scripts
2. Functions in Shell Scripts
- 3. Introduction to grep Commands**
4. Introduction to Regular Expressions
5. Introduction to sed
6. Introduction to AWK Statements

Overview of the grep Command

- global regular expression search and print (grep) is a text search tool that searches for targets line by line based on patterns and displays the results.
- Syntax: grep [*option*] *matching pattern file/data*

```
# View the source file.  
[root@openEuler ~]# cat grep_test.txt  
aaaaaa  
ccccc  
aaaaaa bbbbb  
aa bb  
AAAAAA bbbbb  
[root@openEuler ~]#  
# Display the number of matched lines.  
[root@openEuler ~]# grep -c aaaa grep_test.txt  
2  
[root@openEuler ~]#
```

```
# View the source data.  
[root@openEuler ~]# df -h  
Filesystem      Size  Used Avail Use% Mounted on  
devtmpfs        3.7G   0    3.7G  0% /dev  
/dev/sda2       38G  1.6G  34G  5% /  
/dev/sda1       2.0G  96M  2.0G  5% /boot  
[root@openEuler ~]#  
  
# Display the matched lines.  
[root@openEuler ~]# df -h | grep sda2  
/dev/sda2       38G  1.6G  34G  5% /  
[root@openEuler ~]#
```

- The **grep** command is used to retrieve character string templates in one or more files. The search result is sent to the standard output and does not affect the content of the original file.
- **grep** can be used in shell scripts because **grep** indicates the search status by returning a status value. Return values are used for automated text processing. For example, if the template search is successful, **0** is returned. If the search fails, **1** is returned. If there is no file, **2** is returned.

Overview of the grep Command Options

- Run the **grep -help** command to view all options.
- The **grep** command provides the following types of options:
 - Pattern selection
 - Output control
 - Context control
 - Miscellaneous

```
[root@openEluer ~]# grep --help
Usage: grep [OPTION]... PATTERN [FILE]...
Search for PATTERNS in each FILE...
Example: grep -l 'hello world' menu.h main.c
PATTERNS can contain multiple patterns separated by newlines.

Pattern selection and interpretation:
-E, --extended-regexp      PATTERNS are extended regular expressions
-F, --fixed-strings         PATTERNS are strings
-G, --basic-regexp          PATTERNS are basic regular expressions
-P, --perl-regexp           PATTERNS are Perl regular expressions
-e, --regexp=PATTERNS      use PATTERNS for matching
-f, --file=FILE              take PATTERNS from FILE
-i, --ignore-case            ignore case distinctions in patterns and data
--no-ignore-case             do not ignore case distinctions (default)
-w, --word-regexp            match only whole words
-x, --line-regexp             match only whole lines
-z, --null-data              a data line ends in 0 byte, not newline

Miscellaneous:
-s, --no-messages            suppress error messages
-v, --invert-match           select non-matching lines
-V, --version                 display version information and exit
--help                       display this help text and exit

Output control:
-m, --max-count=NUM          stop after NUM selected lines
-b, --byte-offset              print the byte offset with output lines
-n, --line-number              print line number with output lines
--line-buffered                flush output on every line
-H, --with-filename            print file name with output lines
-h, --no-filename              suppress the file name prefix on output
--label=LABEL                  use LABEL as the standard input file name prefix
-o, --only-matching            show only nonempty parts of lines that match
-q, --quiet, --silent          suppress all normal output
--binary-files=TYPE            assume that binary files are TYPE;
```

grep Output Control Options (1)

```
[root@openEuler ~]# grep -m1 aaa grep_test.txt  
aaaaaa
```

```
[root@openEuler ~]# grep -n aaa grep_test.txt  
1:aaaaaa  
3:aaaaaa bbbbbbb
```

```
[root@openEuler ~]# grep -c aaa grep_test.txt  
2
```

```
[root@openEuler ~]# grep -v a grep_test.txt  
cccccc  
AAAAAA bbbbbbb
```

```
[root@openEuler ~]# grep -o aaaa grep_test.txt  
aaaa  
aaaa
```

View the source file.

```
[root@openEuler ~]# cat grep_test.txt  
aaaaaaaa  
cccccc  
aaaaaaaa bbbbbbb  
aa bb  
AAAAAA bbbbbbb  
[root@openEuler ~]#
```

| Option | Description |
|--------|--|
| -m n | Displays a maximum of n matched lines. |
| -n | Displays the matched lines and line numbers. |
| -c | Displays the number of matched lines. |
| -v | Displays unmatched lines. |
| -o | Displays the matched content. |

grep Output Control Options (2)

```
[root@openEuler ~]# grep -H aaa grep_test.txt  
grep_test.txt:aaaaaa  
grep_test.txt:aaaaaa bbbbbbb  
[root@openEuler ~]#
```

```
[root@openEuler ~]# grep -l aaa grep_test.txt  
grep_test.txt  
[root@openEuler ~]#
```

```
[root@openEuler ~]# grep -L aaa grep_test.txt  
grep_test2.txt  
grep_test2.txt  
[root@openEuler ~]#
```

View the source file.

```
[root@openEuler ~]# cat grep_test.txt  
aaaaaaaa
```

cccccc

aaaaaaaa bbbbbbb

aa bb

AAAAAAA bbbbbbb

```
[root@openEuler ~]# cat grep_test2.txt
```

abcd

acde

```
[root@openEuler ~]#
```

| Option | Description |
|--------|--|
| -H | Displays the matched lines and file names. |
| -l | Displays the matched file names. |
| -L | Displays the unmatched file names. |

grep Context Control Options

```
[root@openEuler ~]# grep -A1 cccc grep_test.txt  
cccccc  
aaaaaa bbbbb  
[root@openEuler ~]#
```

```
[root@openEuler ~]# grep -B1 cccc grep_test.txt  
aaaaaa  
cccccc  
[root@openEuler ~]#
```

```
[root@openEuler ~]# grep -C1 cccc grep_test.txt  
aaaaaa  
cccccc  
aaaaaa bbbbb  
[root@openEuler ~]#
```

View the source file.

```
[root@openEuler ~]# cat grep_test.txt  
aaaaaa  
cccccc  
aaaaaa bbbbb  
aa bb  
AAAAAA bbbbb  
[root@openEuler ~]#
```

| Option | Description |
|--------|---|
| -An | Displays the last <i>n</i> lines that are matched. |
| -Bn | Displays the first <i>n</i> lines that are matched. |
| -Cn | Displays the matched line and <i>n</i> lines before and after it. |

grep Pattern Selection Options

```
[root@openEuler ~]# grep -w aa grep_test.txt  
aa bb  
[root@openEuler ~]#
```

```
[root@openEuler ~]# grep -x aaaaaa grep_test.txt  
aaaaaa  
[root@openEuler ~]#
```

```
[root@openEuler ~]# grep -i aaa grep_test.txt  
aaaaaa  
aaaaaa bbbbb  
AAAAAA bbbbb  
[root@openEuler ~]#
```

```
[root@openEuler ~]# grep -E 'a{3,5}' grep_test.txt  
aaaaaa  
aaaaaa bbbbb  
[root@openEuler ~]#
```

```
# View the source file.  
[root@openEuler ~]# cat grep_test.txt  
aaaaaaaa  
cccccc  
aaaaaaaa bbbbb  
aa bb  
AAAAAA bbbbb  
[root@openEuler ~]#
```

| Option | Description |
|--------|--|
| -w | Matches only whole words. |
| -x | Matches only whole lines. |
| -i | Ignores case distinctions. |
| -E | Supports matching by using extended regular expressions. |

Contents

1. Conditional Judgment in Shell Scripts
2. Functions in Shell Scripts
3. Introduction to grep Commands
- 4. Introduction to Regular Expressions**
5. Introduction to sed
6. Introduction to AWK Statements

Introduction to Regular Expressions

- A regular expression describes a character string-matching pattern that is used to describe and match a series of character strings that comply with a certain rule.
- Supported commands: **locate**, **find**, **vim**, **grep**, **sed**, **awk** and more, for example:
 - grep [0-9a-zA-Z] test.txt
 - sed '/[0-9a-zA-Z]/' test.txt
 - awk '/[0-9a-zA-Z]/' test.txt

Composition of Regular Expressions

- Common characters: Uppercase letters, lowercase letters, digits, punctuations, and other symbols.
 - Example: grep aaa grep_test.txt
- Common metacharacters: Dedicated characters that have special meanings in regular expressions.
 - Example: grep '^a' grep_test.txt
- Extended metacharacters: are used together with escape characters or special options.
 - Example 1: grep 'a\b' grep_test.txt
 - Example 2: grep -E 'a | b' grep_test.txt
- A regular expression consists of three parts: characters, quantity qualifiers, and position qualifiers.

Common Metacharacters in Regular Expressions (1)

```
[root@openEuler ~]# grep 'bbb.' grep_test.txt  
aaaaaa bbbbb  
AAAAAA bbbbb
```

```
[root@openEuler ~]# grep 'bbb*' grep_test.txt  
aaaaaa bbbbb  
aa bb  
AAAAAA bbbbb
```

```
[root@openEuler ~]# grep '^a' grep_test.txt  
aaaaaa  
aaaaaa bbbbb  
aa bb
```

```
[root@openEuler ~]# grep 'b$' grep_test.txt  
aaaaaa bbbbbb  
aa bb  
AAAAAA bbbbbb
```

View the source file.

```
[root@openEuler ~]# cat grep_test.txt  
aaaaaa  
cccccc  
aaaaaa bbbbb  
aa bb  
AAAAAA bbbbb  
[root@openEuler ~]#
```

| Character | Description |
|-----------|--|
| . | Matches any single character except the newline character. |
| * | Indicates that the preceding character appears zero or more consecutive times. |
| ^ | Matches the starting position of the line. |
| \$ | Matches the ending position of the line. |
| ^\$ | Indicates the blank line. |

Common Metacharacters in Regular Expressions (2)

```
[root@openEuler ~]# grep [a] grep_test2.txt  
abcd  
acde  
[root@openEuler ~]#
```

```
[root@openEuler ~]# grep [a-c] grep_test2.txt  
abcd  
acde  
[root@openEuler ~]#
```

```
[root@openeuler ~]# grep [^a-d] grep_test2.txt  
acde  
[root@openeuler ~]#
```

```
[root@openEuler ~]# grep '.*' grep_test2.txt  
abcd  
acde  
[root@openEuler ~]#
```

View the source file.
[root@openEuler ~]# cat grep_test2.txt
abcd
acde
[root@openEuler ~]#

| Character | Description |
|-----------|---|
| [] | Matches a single character or a group of characters. |
| - | Specifies a character range. |
| [^] | Matches characters that do not include a single character or a group of characters. |
| .* | Matches characters of any length. |

Position Qualifier in Regular Expressions

```
[root@openeuler ~]# grep '\<a' grep_test2.txt  
abcd  
acde  
[root@openeuler ~]# grep 'd\>' grep_test2.txt  
abcd  
[root@openeuler ~]# grep '\ba' grep_test2.txt  
abcd  
acde  
[root@openeuler ~]# grep 'd\b' grep_test2.txt  
abcd  
[root@openeuler ~]# grep 'd\B' grep_test2.txt  
acde  
[root@openeuler ~]# grep '\Ba' grep_test2.txt
```

```
# View the source file.  
[root@openEuler ~]# cat grep_test2.txt  
abcd  
acde  
[root@openEuler ~]#
```

| Character | Description |
|-----------|---|
| \< | Word boundary, which matches the position of the beginning of a word. |
| \> | Word boundary, which matches the position of the end of a word. |
| \b | Matches the beginning or end of a word. |
| \B | Matches the position that is not the beginning or end of a word. |

Extended Metacharacters in Regular Expressions (1)

```
[root@openEuler ~]# grep -E 'a+' grep_test2.txt  
abcd  
acde  
[root@openEuler ~]#
```

```
[root@openEuler ~]# grep -E 'g?' grep_test2.txt  
abcd  
acde  
[root@openEuler ~]#
```

```
[root@openEuler ~]# grep -E 'a|c' grep_test2.txt  
abcd  
acde  
[root@openEuler ~]#
```

```
[root@openEuler ~]# grep -E 'a(b|c)' grep_test2.txt  
abcd  
acde  
[root@openEuler ~]#
```

View the source file.
[root@openEuler ~]# cat grep_test2.txt
abcd
acde
[root@openEuler ~]#

| Character | Description |
|-----------|--|
| + | Matches the preceding character one or more times. |
| ? | Matches the preceding character zero or one time. |
| | Indicates or. |
| () | Indicates a capturing group. |

Extended Metacharacters in Regular Expressions (2)

```
[root@openEuler ~]# grep -E 'a{4}' grep_test.txt  
aaaaaa  
aaaaaa bbbbb
```

```
[root@openEuler ~]# grep -E 'a{4,}' grep_test.txt  
aaaaaa  
aaaaaa bbbbb
```

```
[root@openEuler ~]# grep -E 'a{,4}' grep_test.txt  
aaaaaa  
ccccc  
aaaaaa bbbbb  
aa bb  
AAAAAA bbbbb
```

```
[root@openEuler ~]# grep -E 'a{4,6}' grep_test.txt  
aaaaaa  
aaaaaa bbbbb
```

View the source file.

```
[root@openEuler ~]# cat grep_test.txt  
aaaaaa  
ccccc  
aaaaaa bbbbb  
aa bb  
AAAAAA bbbbb  
[root@openEuler ~]#
```

| Character | Description |
|-----------|--|
| {n} | The preceding character repeats <i>n</i> times. |
| {n,} | The preceding character repeats at least <i>n</i> times. |
| {,m} | The preceding character repeats for a maximum of <i>m</i> times. |
| {n,m} | The preceding character repeats <i>n</i> to <i>m</i> times. |

Common Matching Characters in Regular Expressions

```
[root@openEuler ~]# grep -E '\w' grep_test3.txt
abcd
123
[root@openEuler ~]# grep -E '\W' grep_test3.txt
@@
$$
[root@openEuler ~]#
```

```
# View the source file.
[root@openEuler ~]# cat grep_test3.txt
abcd
123
@@
$$
[root@openEuler ~]#
```

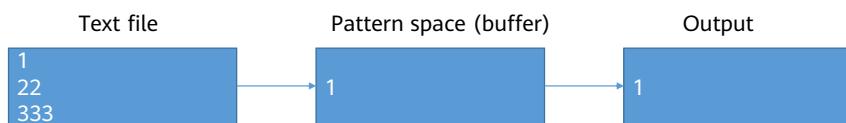
| Character | Description |
|-----------|--|
| \w | Matches a single word character (letters, digits, or underscores). |
| \W | Matches a single non-word character. |

Contents

1. Conditional Judgment in Shell Scripts
2. Functions in Shell Scripts
3. Introduction to grep Commands
4. Introduction to Regular Expressions
- 5. Introduction to sed**
6. Introduction to AWK Statements

Functions and Principles of sed

- sed, short for stream editor, is used for file processing.
- sed processes content line by line. It stores the current line in a temporary buffer, processes the contents in the buffer based on commands, and outputs the results to the screen.
- Syntax:
 - `sed [option] '[locator]command file'`
 - `sed [option] -f script_name.sed file`
 - `./script_name.sed file`



- sed is used to add, delete, modify, and query data within files. It can automatically process one or more files, automate repeated operations, and create conversion programs.
- Line-by-line processing is preferable for processing very large files, as it avoids memory overflow and slow processing if an entire file were loaded into memory at once.
- sed output results can be redirected to the storage side but not back to the original file.

sed Options

- The **sed -h** command displays the options.

```
[root@openEuler ~]# sed '1p' sed.txt
I am Sopia
I am Sopia
[root@openEuler ~]# sed -n '1p' sed.txt
I am Sopia
[root@openEuler ~]# sed -e '1p' -e '1i hello' sed.txt
I am Sopia
hello
I am Sopia
[root@openEuler ~]# sed -i '1i hello world' sed.txt
[root@openEuler ~]# sed -f sed_script.sed sed.txt
hello world
hello world
I am Sopia
[root@openEuler ~]# sed -rn '/l{2}/p' sed.txt
hello wolrd
```

```
[root@openEuler ~]# cat sed.txt
I am Sopia
[root@openEuler ~]# cat sed_script.sed
#!/bin/sed -f
1p
[root@openEuler ~]#
```

| Option | Description |
|--------|-------------------------------------|
| -n | Cancels the default output. |
| -e | Executes multiple editing commands. |
| -i | Modifies the input file. |
| -f | Specifies the sed script. |
| -r/-E | Uses extended regular expressions. |

sed Usage – Line Number Addressing

- Lines can be specified by numbers. Commas (,), wavy signs (~), and plus signs (+) are used to execute various line-based operations.

```
[root@openEuler ~]# sed -n '3p' sed2.txt      # Print line 3.  
333  
[root@openEuler ~]# sed -n '1,3p' sed2.txt    # Print lines 1 to 3.  
1  
22  
333  
[root@openEuler ~]# sed -n '2~2p' sed2.txt   # Print from line 2 with a step of  
2.  
22  
4444  
666666  
[root@openEuler ~]# sed -n '2,+3p' sed2.txt  # Print line 2 and the following  
three lines.  
22  
333  
4444  
555555  
[root@openEuler ~]# sed -n '/33/=' sed2.txt  # Print numbers of the lines that  
contain the specified content.  
3
```

```
[root@openEuler ~]# cat sed2.txt  
1  
22  
333  
4444  
55555  
666666  
[root@openEuler ~]#
```

sed Usage – Regular Expression Addressing

- Uses slashes (/) to enclose the regular expressions. Use the -r option to specify extended regular expressions.

```
[root@openEuler ~]# grep 1 sed2.txt  
1  
[root@openEuler ~]# sed -n '/1/p' sed2.txt  
1  
[root@openEuler ~]# grep -E '[0-9]{3}' sed2.txt  
333  
4444  
55555  
666666  
[root@openEuler ~]# sed -rn '/[0-9]{3}/p' sed2.txt  
333  
4444  
55555  
666666  
[root@openEuler ~]#
```

```
[root@openEuler ~]# cat sed2.txt  
1  
22  
333  
4444  
55555  
666666  
[root@openEuler ~]#
```

sed Usage – Adding, Deleting, Modifying, and Querying Texts

```
[root@openEuler ~]# sed 'ihello' sed2.txt    # Add "hello" before each line.  
hello  
1  
hello  
22  
[root@openEuler ~]# sed '1ihello' sed2.txt   # Add "hello" before the first line.  
hello  
1  
22  
[root@openEuler ~]# sed '1ahello' sed2.txt   # Add "hello" after the first line.  
1  
hello  
22  
[root@openEuler ~]# sed '1d' sed2.txt       # Delete the first line.  
22  
[root@openEuler ~]# sed '1chello' sed2.txt   # Replace the first line with  
"hello".  
hello  
22
```

```
[root@openEuler ~]# cat sed2.txt  
1  
22  
[root@openEuler ~]#
```

| Command | Description |
|---------|---------------------------------|
| p | Prints. |
| i | Inserts before specified lines. |
| a | Inserts after specified lines. |
| d | Deletes specified lines. |
| c | Replaces specified lines. |

sed Usage – Searching and Replacing Texts

- sed *option 's/search pattern/replacement pattern/action' file*
- If the pattern contains slashes (/), custom separators can be used, such as @ or #. & in the replacement pattern is replaced by the text that was matched by the search pattern.

```
[root@openEuler ~]# sed 's/2/hello/' sed2.txt # Replace the first "2" in the line with  
"hello."  
1  
hello2  
/root/  
[root@openEuler ~]# sed 's/2/hello/g' sed2.txt # Replace all occurrences of "2" in the  
line with "hello."  
1  
hellohello  
/root/  
[root@openEuler ~]# sed 's@/root/@hello@' sed2.txt # Replace "/root/" with "hello."  
1  
22  
hello  
[root@openEuler ~]# sed 's/2/&hello/' sed2.txt  
1  
2hello2  
/root/
```

```
[root@openEuler ~]# cat sed2.txt  
1  
22  
/root/  
[root@openEuler ~]#
```

sed Usage – File Operations

```
# Insert the content of sed.txt after the first line of
# sed2.txt.
[root@openEuler ~]# sed '1r sed.txt' sed2.txt
1
hello wolrd
I am Sopia
I am 3 years old
22
/root/
# Save the first line of sed2.txt to sed.txt.
[root@openEuler ~]# sed '1w sed.txt' sed2.txt
1
22
/root/
[root@openEuler ~]# cat sed.txt
1
hello wolrd
I am Sopia
I am 3 years old
```

```
[root@openEuler ~]# cat sed2.txt
1
22
/root/
[root@openEuler ~]# cat sed.txt
hello wolrd
I am Sopia
I am 3 years old
[root@openEuler ~]#
```

| Command | Description |
|---------|----------------------------------|
| r | Reads content from another file. |
| w | Saves content as another file. |

sed Usage – Grouping

- Braces ({}) in sed commands are used for two purposes:
 - Contain a range within another range (nesting).
 - Perform multiple commands or actions on the same range (grouping).

```
[root@openEuler ~]# cat file.txt
philadephia 1
dc 2
start 3
dc 4
philly 5
dc 6
nyc 7
dc 8
end 9
dc 10
```

```
[root@openEuler ~]# sed '/start/,/end/{ /philly/,/nyc/s/dc/district/g }'
file.txt
philadephia 1
dc 2
start 3
dc 4
philly 5
district 6
nyc 7
dc 8
end 9
dc 10
```

sed Usage – Executing Expressions Serially

- Multiple expressions can be executed serially in either of the following ways:
 - sed '*expression; expression*'
 - Use the **-e** option to execute multiple commands on the same line. The order of commands affects the results.

```
# cat phone.list
Terrell, Terry 617-7989
Franklin, Francis 704-3876
Patterson, Pat 614-6122
Robinson, Robin 411-3745
Christopher, Chris 305-5981
Martin, Marty 814-5587
Llewellyn, Lynn 316-6221
Jansen, Jan 903-3333
Llewellyn, Lee 817-8823
```

```
# sed '/[1-4]-/s/$/ (Bldng 1) /;/[5-9]-/s/$/ {Bldng 2} /' phone.list
Terrell, Terry 617-7989 {Bldng 2}
Franklin, Francis 704-3876 (Bldng 1)
Patterson, Pat 614-6122 (Bldng 1)
Robinson, Robin 411-3745 (Bldng 1)
Christopher, Chris 305-5981 {Bldng 2}
Martin, Marty 814-5587 (Bldng 1)
Llewellyn, Lynn 316-6221 {Bldng 2}
Jansen, Jan 903-3333 (Bldng 1)
Llewellyn, Lee 817-8823 {Bldng 2}
```

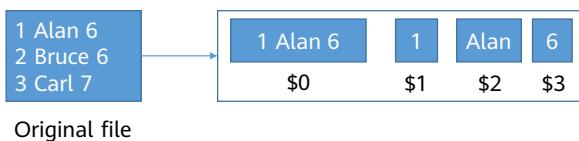
Contents

1. Conditional Judgment in Shell Scripts
2. Functions in Shell Scripts
3. Introduction to grep Commands
4. Introduction to Regular Expressions
5. Introduction to sed
- 6. Introduction to AWK Statements**

Introduction to AWK

- AWK is a programming language used to process text or data in Linux and Unix, with input sources including standard input, files, and command outputs.
- AWK progressively scans and searches for lines that match specified patterns and then processes these lines. It supports conditional statements, as well as for and while loops.
- Syntax:

▫ `awk option 'mode{action}' file`



```
[root@openEuler ~]# cat awk.txt
1 Alan 6
2 Bruce 6
3 Carl 7
[root@openEuler ~]# awk '{print $2}' awk.txt
Alan
Bruce
Carl
[root@openEuler ~]#
```

- AWK reads one line at a time and assigns it to the built-in variable **\$0**. Each line, also called a record, ends with a newline character **RS** (short for record separator).
- Each line is split into fields by a separator (space or tab by default), and each field is stored in a numbered variable, starting from **\$1**.
- AWK uses the **print** function to print fields. The printed fields are separated by spaces, because there is a comma between **\$1** and **\$3**. A comma is a special character that maps to another built-in variable called Output Field Separator (OFS). The default value of OFS is a space.
- After processing a line, AWK reads another line from the file and stores it in **\$0**, overwriting the existing value. Then AWK splits the new line into fields and processes them.

AWK Patterns and Actions

- Syntax:
 - `awk -F separator 'pattern {action}pattern {action}pattern{action}' file`
 - `awk -F separator 'pattern {action} pattern {action} pattern{action}' file`
 - `awk -F separator 'pattern {action};pattern {action};pattern{action}' file`
- A pattern and an action are combined to form a statement. The entire statement is enclosed in single quotation marks ('). Statements can be placed next to each other or separated by spaces or semicolons (;). This is not needed when there is only one statement.
- Both patterns and actions are optional. When the pattern is omitted, the statement matches all lines by default. When the action is omitted, the default action is to print the entire line.
- Actions are enclosed in braces while patterns are not.

AWK Pattern Overview

- Patterns determine when and what actions are triggered.
- A pattern can be one of the following.

| Pattern | Description |
|---|--|
| BEGIN; END | Special pattern |
| /regular expression/ | Regular expression |
| relational expression | Relational expression |
| pattern && pattern; pattern pattern; ! pattern | Logical expression |
| pattern ? pattern : pattern | Ternary expression |
| (pattern) | Constant expression. The action is executed when the expression is true. |
| pattern1, pattern2 | Range pattern |

- If the pattern is omitted, the action is executed on all records.

Common AWK Patterns – Regular Expressions

- Content matching: `~/content/` or `!~/content/`
- Enclose regular expressions in slashes (/).

```
[root@openEuler ~]# cat awk.txt
1 Alan 6
2 Bruce 6
3 Carl 7
[root@openEuler ~]#
```

```
# Print the line that is separated by colons (:) or spaces, has a line number greater than 1,
# and contains 6 (the second line).
[root@openEuler ~]# awk -F'[ :]' 'NR>1 && $0 ~ /6/{print $0}' awk.txt
2 Bruce 6
[root@openEuler ~]#
```

Common AWK Patterns – Constant Expressions

- A constant can be a string or numeric constant. A string constant is enclosed in quotation marks while a numeric constant is not.
- A non-zero numeric constant indicates successful matching, that is, the pattern is true. Otherwise, the pattern is false.
- A non-empty string constant indicates successful matching, that is, the pattern is true. Otherwise, the pattern is false.
- awk ' true {*action to execute*}; false {*action not to execute*}';

- A string constant is enclosed in quotation marks. For example, the number 0 differs from the string "0". The number 0 indicates false but the string "0" indicates true because it is not empty.

AWK Field Separators

- The field separator can be specified the **-F** option or the **FS** variable in the **BEGIN**

```
# Specify the input field separator.  
[root@openEuler ~]# awk -F: '{print $2}' awk2.txt  
Alan  
Bruce  
Carl  
[root@openEuler ~]# awk 'BEGIN{FS=":"};{print $2}' awk2.txt  
Alan  
Bruce  
Carl
```

| Variable | Description |
|----------|---|
| FS | Input field separator. |
| OFS | Output field separator. The default value is a space. |

```
# Specify the output field separator.  
[root@openEuler ~]# awk -F: '{print $2":"$3}' awk2.txt  
Alan:6  
Bruce:6  
Carl:7  
[root@openEuler ~]# awk -F: 'BEGIN{OFS=":"};{print $2,$3}' awk2.txt  
Alan;6  
Bruce;6  
Carl;7
```

```
[root@openEuler ~]# cat awk2.txt  
1:Alan:6  
2:Bruce:6  
3:Carl:7  
[root@openEuler ~]#
```

AWK Record Separators

```
# Specify the input record separator.  
[root@openEuler ~]# awk 'BEGIN{RS="."};{print $0}' awk2.txt  
1  
Alan  
6  
2  
Bruce  
6  
3  
Carl  
7
```

```
# Specify the output record separator.  
[root@openEuler ~]# awk -F: 'BEGIN{ORS=","};{print $2,$3}'  
awk2.txt  
Alan 6;Bruce 6;Carl 7;
```

```
[root@openEuler ~]# cat awk2.txt  
1:Alan:6  
2:Bruce:6  
3:Carl:7  
[root@openEuler ~]#
```

| Variable | Description |
|----------|--|
| RS | Input record separator. The default value is a newline character. |
| ORS | Output record separator. The default value is a newline character. |

AWK Action Overview

- Actions are separated by newline characters or semicolons and enclosed in braces.
- Actions include built-in functions, variable or array assignment, input and output statements, and control statements (conditional and loop statements).
- If the action is omitted, the default action is executed, that is, **{print \$0}**, which prints all matched records.

Common AWK Actions – Built-in Functions

- AWK functions are classified into built-in functions and user-defined functions.

| Classification | Type | Name |
|------------------------|--------------------------------|---|
| Built-in functions | Arithmetic functions | atan2(y,x), cos(expr), exp(expr), int(expr), log(expr), rand(), sin(expr), sqrt(expr), srand([expr]) |
| | String functions | asort(s [, d]), asorti(s [, d]), gensub(r, s, h [, t]), gsub(r, s [, t]), index(s, t), length([s]), match(s, r [, a]), split(s, a [, r]), sprintf(fmt, expr-list), strtonum(str), sub(r, s [, t]), substr(s, i [, n]), tolower(str), toupper(str) |
| | Time functions | mktime(datespec), strftime([format [, timestamp[, utc-flag]]]), systime() |
| | Bitwise functions | and(v1, v2), compl(val), lshift(val, count), or(v1, v2), rshift(val, count), xor(v1, v2) |
| | Internationalization functions | bindtextdomain(directory [, domain]), dcgettext(string [, domain [, category]]), dcngettext(string1 , string2 , number [, domain [, category]]) |
| User-defined functions | User-defined functions | function name(parameter list) { statements } |

AWK Actions – Function Example

- The following is an example of the string replacement function:

```
#cat employees
Mary Adams    5346  11/4/63
28765

Sally Chang   1654  7/22/54      650000
Billy Black   1683  9/23/44      336500

#awk '{sub("l","8"); print}' employees
# Replace the first occurrence of l found in each line with 8.
Mary Adams    5346  11/4/63      28765
Sa8ly Chang   1654  7/22/54      650000
Bi8ly Black   1683  9/23/44      336500

# awk '{sub("l","8",$2); print}' employees
# Replace the first occurrence of l found in the second
# column of each line with 8.
Mary Adams    5346  11/4/63      28765
Sally Chang   1654  7/22/54      650000
Billy B8ack   1683  9/23/44      336500
```

```
#awk '{gsub("l","8"); print}' employees
# Replace all occurrences of l found in each line with 8.
Mary Adams    5346  11/4/63      28765
Sa88y Chang   1654  7/22/54      650000
Bi88y B8ack   1683  9/23/44      336500

#awk '{gsub("l","8",$1); print}' employees
# Replace all occurrences of l found in the first column
# of each line with 8.
Mary Adams    5346  11/4/63      28765
Sa88y Chang   1654  7/22/54      650000
Bi88y Black   1683  9/23/44      336500
```

Common AWK Actions – Built-in Variables

```
[root@openEuler ~]# awk '/Alan/{print $0}' awk.txt
1 Alan 6
[root@openEuler ~]# awk '/Alan/{print $2}' awk.txt
Alan
[root@openEuler ~]# awk '/Alan/{print $1,$2}' awk.txt
1 Alan
[root@openEuler ~]# awk '/Alan/{print NF}' awk.txt
3
[root@openEuler ~]# awk '/Alan/{print $NF}' awk.txt
6
[root@openEuler ~]# awk 'NR==2{print $2}' awk.txt
Bruce
[root@openEuler ~]#
```

```
[root@openEuler ~]# cat awk.txt
1 Alan 6
2 Bruce 6
3 Carl 7
```

| Variable | Description |
|---------------|--|
| \$0 | All records of the line being processed |
| \$1,\$2...\$n | Separated fields in each line of a file |
| NF | Number of fields (columns) in the current record |
| \$NF | Last column |
| FNR/NR | Line number |

Common AWK Actions – User-Defined Variables

- awk '{*action*'}, *variable_name*=*variable_value*, *file*
- Example: **awk '{print a, b}' a=111 b=222 filename**
 - The variables must be declared before the file name. Otherwise, they cannot be used.
 - **BEGIN{}** cannot use user-defined variables.
 - **-v** defines variables and assigns values to them. The dollar sign (\$) is not required when you use the variables.

```
[root@openEuler ~]# awk -v a="Name:" '{print a$2}' awk.txt
Name:Alan
Name:Bruce
Name:Carl
[root@openEuler ~]#
```

AWK Usage Scenario – Formatted Output

- **print** is similar to **echo**. By default, **print** adds newline characters at the end of the output.

```
[root@openEuler ~]# awk '{print "No:"$1,"Name:"$2,"Age:"$3}' awk.txt
No:1 Name:Alan Age:6
No:2 Name:Bruce Age:6
No:3 Name:Carl Age:7
```

```
[root@openEuler ~]# cat awk.txt
1 Alan 6
2 Bruce 6
3 Carl 7
[root@openEuler ~]#
```

- **printf** is similar to **echo -n**. By default, **printf** does not add newline characters at the end of the output. **printf** uses C-style format specifiers.

- $\%[flags][minimum_width][.precision]converter$

```
[root@openEuler ~]# awk '{printf "%5d%-10s%-5d\n",$1,$2,$3}' awk.txt
1Alan    6
2Bruce   6
3Carl    7
[root@openEuler ~]#
```

- Converters:

- **c**: ASCII character (first character of a string)
- **s**: string
- **d, i**: decimal integer
- **u**: unsigned decimal integer
- **x**: unsigned hexadecimal integer, with **a** to **f** representing 10 to 15
- **X**: unsigned hexadecimal integer, with **A** to **F** representing 10 to 15
- **o**: unsigned octal integer
- **e, E**: floating-point number in scientific notation (E notation)
- **f, F**: floating-point number
- **g**: equal to **%e** or **%f** depending on which is shorter, with trailing zeros removed
- **G**: equal to **%E** or **%F** depending on which is shorter, with trailing zeros removed
- **%%**: literal percent sign (%)

- Flags:
 - -: aligns output to the left.
 - (space): prints a space before a positive number and a minus sign before a negative number.
 - +: always prints a positive or minus sign before a number.
 - #:
 - Prefixes a **%o** number with **0**.
 - Prefixes a **%x** number with **0x**.
 - Prefixes a **%X** number with **0X**.
 - Adds decimal points to **%e**, **%E**, and **%f** numbers.
 - Retains trailing zeros for **%g** and **%G** numbers.
 - 0: pads a printed value with zeros instead of spaces.

Precision of printf

- When the precision specifies the number of digits after the decimal point, the default conversion format for numbers is determined by the system variable **CONVFMT** (or **OFMT** in some systems), which defaults to **%6g**.

| Converter | Description | Example |
|------------------------|---|---|
| %d, %i, %o, %u, %x, %X | Minimum number of digits. Numbers with fewer digits than the precision are padded with leading zeros. | <pre>#echo "35632" awk '{ printf " %.8d \n",\$1}' 00035632 #echo "3563287654" awk '{ printf " %.8d \n",\$1}' 3563287654 </pre> |
| %e, %E | Minimum number of digits. Numbers with fewer digits than the precision are padded with trailing zeros. | <pre>#echo "3563287654" awk '{ printf " %.8e \n",\$1}' 3.56328765e+09 #echo "35632" awk '{ printf " %.8e \n",\$1}' 3.56320000e+04 </pre> |
| %f | Number of decimal places. | <pre>#echo "1.2456" awk '{ printf "%.10f\n",\$1}' 1.2456000000</pre> |
| %g, %G | Maximum number of significant digits. | <pre>#echo "35632" awk '{ printf " %.8g \n",\$1}' 35632 #echo "3563287654" awk '{ printf " %.8g \n",\$1}' 3.5632877e+09 </pre> |
| %s | Number of characters. | <pre>#echo "helloworld" awk '{ printf " %.8s \n",\$1}' helloworld </pre> |

AWK Usage Scenario – Control Statements

- Conditional expressions in AWK control statements are enclosed in parentheses.

```
# Conditional statement
[root@openEuler ~]# cat awk.txt
1 Alan 6
2 Bruce 6
3 Carl 7
[root@openEuler ~]# awk '{if($3==6){print $2}}' awk.txt
Alan
Bruce
[root@openEuler ~]#
```

```
# Loop statement
[root@openEuler ~]# awk 'BEGIN{for(i=1;i<10;i++){printf "***";printf "\n"};{print $2};
END{i=1;while(i<10){printf "***";i++;printf "\n"}}' awk.txt
*****
Alan
Bruce
Carl
*****
[root@openEuler ~]#
```

AWK Script Mode

- You can save a list of commands in a script file. Commands are separated by newlines or semicolons (;) and are not enclosed in quotation marks.
- awk -f awk.sh *file*
- ./awk.sh *file*

```
[root@openEuler ~]# cat awk2.txt
1:Alan:6
2:Bruce:6
3:Carl:7
[root@openEuler ~]# cat awk.sh
#!/bin/awk -f
BEGIN{FS=":";a="No";b="Name";c="Age";printf "%-5s%-10s%-5s\n",a,b,c}
NR>1{printf "%-5d%-10s%-5d\n",$1,$2,$3}
[root@openEuler ~]#
```

```
[root@openEuler ~]# awk -f awk.sh awk2.txt
No Name    Age
2  Bruce   6
3  Carl    7
[root@openEuler ~]#
```

Quiz

1. In the conditional judgment of openEuler, which of the following syntax must contain a space before and after an expression?
 - A. test
 - B. []
 - C. [[]]
 - D. ()()
2. The sed **-n** option displays the matched lines and line numbers.
 - A. True
 - B. False

- BC
- B

Summary

- This course describes the advanced functions of shell, including conditional judgment, functions, regular expressions, and common Linux command tools grep, sed, and AWK.
 - grep is used for data retrieval.
 - sed is used for data modification.
 - AWK is used for data slicing and formatting, and has the most complex functions.

Acronyms and Abbreviations

| Acronym/Abbreviation | Full Spelling |
|----------------------|--|
| grep | global regular expression search and print |

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and
organization for a fully connected,
intelligent world.

Copyright©2024 Huawei Technologies Co., Ltd.
All Rights Reserved.

The information in this document may contain predictive
statements including, without limitation, statements regarding
the future financial and operating results, future product
portfolio, new technology, etc. There are a number of factors that
could cause actual results and developments to differ materially
from those expressed or implied in the predictive statements.
Therefore, such information is provided for reference purpose
only and constitutes neither an offer nor an acceptance. Huawei
may change the information at any time without notice.



openEuler OS Security Hardening



Foreword

- The operating system (OS) serves as a critical foundation for the Internet, functioning as a vital entity for generating, managing, and preserving essential data. openEuler is a free and open source Linux distribution, offering a robust and thriving ecosystem along with community backing. Given its significance, safeguarding its security is paramount.
- This course delves into OS security, including security hardening policies, network security, and SELinux access control.

Objectives

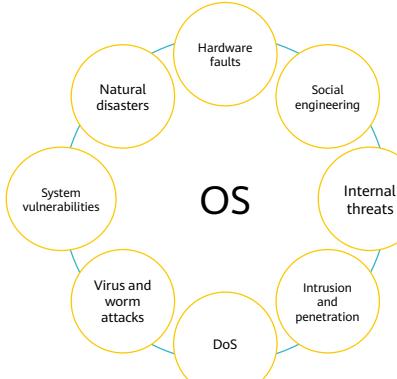
- Upon completion of this course, you will master:
 - openEuler system hardening policies
 - Netfilter architecture and principles
 - Syntax rules of iptables
 - Syntax rules of firewalld
 - SELinux mandatory access control (MAC) policies

Contents

- 1. OS Security Hardening**
2. OS Network Security
3. SELinux Access Control

Major OS Threats

- The operating system (OS) serves as the core of the information system security, managing hardware and software resources and ensuring data integrity, confidentiality, availability, and controllability. Without security protection, hackers and virus attacks can penetrate vulnerabilities across system layers.



- Natural disasters like earthquakes, thunderstorms, and fires, along with hardware faults, are the major areas that affect physical security. These incidents directly compromise the availability of information and disrupt service continuity, ultimately preventing the delivery of services to authorized users.
- Social engineering is a method of deceiving and harming victims through psychological weakness such as instinct, curiosity, trust, and greed. In recent times, social engineering tactics have been instrumental in numerous network breaches, representing a significant threat to data protection.
- Most internal threats are information leakage and damage caused by deliberate misuse of privileges. The primary motivations for deliberate internal incidents are as follows:
 - A desire for financial gains
 - Dissatisfaction with company policies or leadership
 - Personal life discontent leading to atypical conduct in the workplace
- Hackers frequently leverage existing system vulnerabilities and bugs to initiate attacks, which can include aiding in social engineering schemes. Once they infiltrate the system, Trojan horses or viruses lay dormant in the system, to eventually gain control to the system, where it can disclose and temper with core

information.

- Denial of Service (DoS) attacks and distributed denial of service (DDoS) attacks can disrupt service operations of information systems.

OS Security Requirements

- OS security is guaranteed by its own security configurations, security software, and third-party security devices.
- *Information Security Technology - Security Techniques Requirement for Operating System* (GB/T 20272-2019):
 - Class 1: User self-protection
 - Class 2: System audit protection
 - Class 3: Security label protection
 - Class 4: Structuralized protection
 - Class 5: Access authentication protection

Each class contains

Security function requirements
Self-security requirements
Security assurance requirements

- (Chinese version)
<http://c.gb688.cn/bzgk/gb/showGb?type=online&hcno=C372DE6A16690EDB2A8F1A014DA221D6>
- (English version) <https://www.chinesestandard.net/PDF.aspx/GBT20272-2019>

Security Hardening Overview

- Security hardening is a proactive strategy that mitigates potential risks by refining OS configurations, drawing on past insights to prevent future incidents. It helps build a dynamic security system for product security and improves product competitiveness.
- openEuler system hardening includes the following five parts:
 - Kernel parameters
 - Authorization and authentication
 - Accounts and passwords
 - File permissions
 - System services

- Security hardening is a proactive strategy that mitigates potential risks by refining OS configurations, drawing on past insights to prevent future incidents.

Overview of Kernel Parameters

- Kernel parameters determine the status of configurations and application permissions. The kernel provides configurable system control options which can be fine-tuned, to improve OS security.
- Kernel parameters can be read and temporarily modified by running the **sysctl** command, and are stored in the **/proc/sys** directory.
- You can modify the **/etc/sysctl.conf** file to permanently retain kernel parameters.

Kernel Parameter Configuration Tool – `sysctl`

- Run the `sysctl` command to read or write kernel parameters during system running.
- Syntax:
 - `sysctl [options] [variable[=value]] [...]`
- Command options
 - `-a, --all`: displays all kernel parameters.
 - `-w, --write`: temporarily modifies a specified kernel parameter.
 - `-p [FILE]`: loads kernel parameters from a file. By default, kernel parameters are loaded from `/etc/sysctl.conf`.
 - `-n, --values`: displays the values of kernel parameters.
 - `-N, --names`: displays the names of kernel parameters.
 - `-e, --ignore`: ignores unknown parameter errors.

```
[root@Nginx1 ~]# sysctl -a | wc -l
993
[root@Nginx1 ~]# sysctl -p
kernel.sysrq = 0
net.ipv4.ip_forward = 0
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.default.send_redirects = 0
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.default.accept_redirects = 0
net.ipv4.conf.all.secure_redirects = 0
net.ipv4.conf.default.secure_redirects = 0
net.ipv4.icmp.echo_ignore_broadcasts = 1
net.ipv4.icmp_ignore_bogus_error_responses = 1
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.rp_filter = 1
net.ipv4.tcp_syncookies = 1
kernel.dmesg_restrict = 1
net.ipv6.conf.all.accept_redirects = 0
net.ipv6.conf.default.accept_redirects = 0
net.ipv4.conf.all.arp_ignore = 1
net.ipv4.conf.all.arp_announce = 2
net.ipv4.conf.ens192.arp_ignore = 1
net.ipv4.conf.ens192.arp_announce = 2
```

Kernel Parameter Hardening Policy (1)

- Common network hardening policies

| Hardening Item | Description | Recommended Value | Hardened in openEuler |
|---|--|-------------------|-----------------------|
| net.ipv4.icmp_echo_ignore_broadcasts | Ignores ICMP broadcast packets. | 1 | Yes |
| net.ipv4.conf.all.rp_filter | Verify reverse route. | 1 | Yes |
| net.ipv4.conf.default.rp_filter | | 1 | Yes |
| net.ipv4.ip_forward | Forwards data packets. | 0 | Yes |
| net.ipv4.conf.all.accept_source_route | Prohibit a packet sender from specifying the sending and returning path. | 0 | Yes |
| net.ipv4.conf.default.accept_source_route | | 0 | Yes |
| net.ipv4.conf.all.accept_redirects | ICMP redirection | 0 | Yes |
| net.ipv4.conf.default.accept_redirects | | 0 | Yes |
| net.ipv6.conf.all.accept_redirects | | 0 | Yes |
| net.ipv6.conf.default.accept_redirects | | 0 | Yes |

- **icmp_echo_ignore_broadcasts**: prevents ICMP amplification attacks from occupying available bandwidth of the target system and denying communication.
- **rp_filter**: After a NIC receives a data packet, it exchanges the source address with the destination address to search for the reverse route egress, and then filters data packets based on the reverse route egress. When the value of **rp_filter** is **1**, the reverse route egress must be the same as the ingress NIC of data packets; otherwise, data packets will be discarded. If the value is **2**, the reverse route must be reachable; otherwise, data packets will be discarded. It can reduce DDoS attacks by verifying the reverse path of data packets. If the reverse path is inappropriate, data packets are directly discarded to prevent system resource consumption.
- **ip_forward**: data packet forwarding.
- **accept_source_route**: The source route information specifies the transmission path of data packets. Attackers can forge some valid IP addresses to access the network.
- **accept_redirects**: ICMP redirection information is real-time route information provided by a router for a host. When a host receives ICMP redirection information, it updates the route table accordingly. In the absence of validity checks, however, this process is vulnerable to attackers sending false ICMP redirection messages to the target host, prompting it to alter its route table.

Attackers can change the host gateway to their own machines, by intercepting the host data packets, paving the way for further intrusions like man-in-the-middle or DoS attacks.

Kernel Parameter Hardening Policy (2)

| Hardening Item | Description | Recommended Value | Hardened in openEuler |
|--|---|-------------------|-----------------------|
| net.ipv4.conf.all.send_redirects | Disable to send ICMP redirection packets to other hosts. | 0 | Yes |
| net.ipv4.conf.default.send_redirects | This option is enabled only when a host serves as a route. | 0 | Yes |
| net.ipv4.icmp_ignore_bogus_error_responses | Ignores forged ICMP packets, without record them in logs. | 1 | Yes |
| net.ipv4.tcp_syncookies | Enables TCP-SYN cookie protection. | 1 | Yes |
| net.ipv4.conf.all.secure_redirects | Receives ICMP redirection messages from any host. | 0 | Yes |
| net.ipv4.conf.default.secure_redirects | Receives ICMP redirection messages from gateways in the default gateway list. | 0 | Yes |

- Other kernel hardening policies

| Hardening Item | Description | Recommended Value | Hardened in openEuler |
|--------------------------------|--|-------------------|-----------------------|
| kernel.dmesg_restrict | Allows only administrator to view the dmesg information. | 1 | Yes |
| kernel.sched_autogroup_enabled | Prohibits the kernel from automatically grouping and scheduling threads. | 0 | No |
| kernel.sysrq | Disables the magic key. | 0 | Yes |

- **net.ipv4.icmp_ignore_bogus_error_responses:** saves drive space.
- **net.ipv4.tcp_syncookies:** SYN is a DoS attack that forces system restart by occupying system resources. TCP-SYN cookie protection should be enabled for the purpose of security.
- **kernel.sched_autogroup_enabled:** determines whether the kernel automatically groups and schedules threads. When enabled, scheduling groups compete for time slices, and threads in a scheduling group compete for time slices allocated to the scheduling group. But this option is disabled according to the hardening policy.
- **kernel.sysrq:** prevents commands from being directly sent to the kernel, to avoid adverse impacts on the system impact and enhance kernel security.

Authorization and Authentication Hardening Solution

- Common system hardening methods by using authorization and authentication:
 - Configure a warning for remote network access.
 - Disable system restart using **Ctrl+Alt+Del**.
 - Set the automatic timeout for terminals.
 - Set the default user's umask value to **0077**.
 - Set the encryption password of GRUB2.

Authorization and Authentication – Warning for Remote Network Access

- The warning is used to warn users before logging in to a system, indicating that potential attackers may be punished for illegal intrusion. 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. (It is configured in openEuler by default.)

Authorized users only. All activities may be monitored and reported.

- Authorization and authentication

Authorization and Authentication – Disabling System Restart by Pressing 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.
- Detailed operations are as follows:
 - Delete the two ctrl-alt-del.target files.

```
rm -f /etc/systemd/system/ctrl-alt-del.target  
rm -f /usr/lib/systemd/system/ctrl-alt-del.target
```
 - Change **#CtrlAltDelBurstAction=reboot-force** to **CtrlAltDelBurstAction=none** in the **/etc/systemd/system.conf** file.
 - Restart systemd for the modification to take effect.

```
systemctl daemon-reexec
```

Authorization and Authentication – Automatic Logout Time of Terminals

- Unattended devices are prone to listening or attacks. Therefore, you are advised to configure terminals to ensure that they automatically close after a period of inactivity.
- 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`

Authorization and Authentication - Setting the Default User's umask Value to 0077

- The umask value is used to set the default permission on a new file or directory. If the umask value is too small, the permission of group users or other users will be too high, which threatens system security. 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**.
- Procedure
 - Add **umask 0077** to the **/etc/bashrc** file and all files in **/etc/profile.d/**.

```
echo "umask 0077" >> $FILE
```
 - Set the owner and group of the **/etc/bashrc** file and all files in **/etc/profile.d/** to **root**.

```
chown root.root $FILE
```

Authorization and Authentication – GRUB2 Encryption Password

- GRand Unified Bootloader (GRUB) is a boot manager for OSs such as Windows and Linux. GRUB2 is a later version of GRUB.
- You can modify the startup parameters of the system on the GRUB2 GUI. To ensure that the system startup parameters are modified with authorization, encrypt the GRUB2 GUI so that parameters can be modified only when you enter the correct GRUB2 password.

- The default password of GRUB2 is **openEuler#12**. You can run **grub2-mkpasswd-pbkdf2** to generate a new encrypted

```
[root@openEuler ~]# grub2-mkpasswd-pbkdf2
Enter password:
Reenter password:
PBKDF2 hash of your password is
grub.pbkdf2.sha512.10000.5A45748D892672FDA02DD3B6F7AE390AC6E6D532A600D4AC477D25C7D087644697D8A0894DFED9D86DC2A27F4E01D9
25C46417A225FC099C12DBD3D7D49A7425.2BD2F5BF4907DCC389CC5D165DB85CC3E2C94C8F9A30B01DACA9CD552B731BA1DD3B7CC2C765704
D55B8CD962D2AEF19A753CBE9B8464E2B1EB39A3BB4EAB08
```

- Use the vi tool to open **grub.cfg** and add the following fields to the beginning of the file:

```
set superusers="root"
password_pbkdf2 root
grub.pbkdf2.sha512.10000.5A45748D892672FDA02DD3B6F7AE390AC6E6D532A600D4AC477D25C7D087644697D8A0894DFED9D86DC2A27F4E01D9
25C46417A225FC099C12DBD3D7D49A7425.2BD2F5BF4907DCC389CC5D165DB85CC3E2C94C8F9A30B01DACA9CD552B731BA1DD3B7CC2C765704
D55B8CD962D2AEF19A753CBE9B8464E2B1EB39A3BB4EAB08
```

- 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.
- The path of the **grub.cfg** file varies in different modes. For x86, the path is **/boot/efi/EFI/openEuler/grub.cfg** in UEFI mode and **/boot/grub2/grub.cfg** in legacy BIOS mode. For AArch64, the path is **/boot/efi/EFI/openEuler/grub.cfg**.
 - 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.

Resetting the Root Password in Single - User Mode

- If the administrator loses the root password and fails to log in the OS, reset the root password in single-user mode as follows:
 - Before the system starts, press **e** to edit the startup item and add **rd.break** to the end of the line where the specified kernel is located.

```
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
```
 - Press **Ctrl+x** to continue the startup.
 - Launch the shell command line and run **mount -o remount, rw /sysroot** to mount **/sysroot** in read/write mode.
 - Run **chroot /sysroot** to set **/sysroot** to **/**.
 - Run **echo "Huawei@123" | passwd --stdin root** to reset the password. In the command, **Huawei@123** is the new password.
 - Run **touch /.autorelabel** to add a label to the file.
 - Enter **exit** twice to exit and restart the system.

- openEuler startup process:
 - A power-on self-check is performed during system firmware running.
 - The system firmware selects bootable devices in sequence and searches for the master boot record (MBR) of all devices.
 - The system firmware reads the boot loader on the device and transfers the system control to it.
 - The boot loader selects the kernel to be booted based on **grub.cfg** and loads the kernel and initramfs to the memory.
 - The boot loader transfers the system permission to the kernel. The kernel searches for the hardware driver from initramfs and initializes the hardware.
 - Systemd is started and the root file system in the drive is mounted to **/sysroot**.
 - The kernel switches the root file system from initramfs to the root file system in **/sysroot**. Then, systemd is executed again.
 - Systemd finds the default startup target and completes the startup.

Account and Password Hardening Solution

- The following methods are commonly used to harden the system using accounts and passwords:
 - Shield system accounts.
 - Restrict accounts that are allowed to use **su** commands.
 - Set password complexity.
 - Set a password validity period.
 - Set password encryption algorithms.
 - Lock an account after three login failures.
 - Harden **su** commands.

Account Shielding and Restricting for su Commands

- Shield system accounts.
 - Accounts are classified into user accounts and system accounts. The latter can only be used inside a system and cannot be used to log in to the system or perform other operations. Therefore, system accounts need to be shielded.
 - For example, change **shell** of the system account to **/sbin/nologin**.

```
usermod -L -s /sbin/nologin systemaccount
```
- Restrict accounts that are allowed to use **su** commands.
 - **su** commands are used to switch between accounts. For security purpose, it is necessary to control permissions that can 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
```

- Accounts and passwords
- **systemaccount**: system account (daemon, dhcpcd, httpd).
- **use_uid**: UID based on the current account.

Setting Password Complexity

- You can set password complexity by modifying the corresponding configuration file as required.
- The password complexity is implemented by the **pam_pwquality.so** and **pam_pwhistory.so** modules in the **/etc/pam.d/password-auth** and **/etc/pam.d/system-auth** files.

- You can set password complexity by modifying the corresponding configuration file as required.

Example for Password Complexity

- To ensure OS security, passwords must meet the following requirements:
 - Contain at least eight characters.
 - Contain at least three types of the following characters:
 - At least one lowercase letter
 - At least one uppercase letter
 - At least one digit
 - Space or at least one of the following special characters: `~!@#\$%^&*()_-+=|[{}];:"<>/?
 - Cannot be the same as the account name or the account name spelled backward.
 - Be different from any of the last five passwords that have been used.

| | | |
|----------|-----------|--|
| password | requisite | pam_pwquality.so minlen=8 minclass=3 enforce_for_root try_first_pass local_users_only retry=3 dcredit=0 ucredit=0 lcredit=0 ocredit=0 pam_pwhistory.so use_authtok remember=5 enforce_for_root |
| password | required | |

| pam_pwquality.so | |
|------------------|--|
| Parameter | Description |
| minlen=8 | A password must contain at least eight characters. |
| minclass=3 | A password contains at least any three of the following: uppercase letters, lowercase letters, digits, and special characters. |
| ucredit=0 | A password can contain any number of uppercase letters. |
| lcredit=0 | A password can contain any number of lowercase letters. |
| dcredit=0 | A password can contain any number of digits. |
| ocredit=0 | A password can contain any number of special characters. |
| retry=3 | Each password change allows a maximum of three attempts. |
| enforce_for_root | This parameter is also effective for account root. |

| pam_pwhistory.so | |
|------------------|--|
| Parameter | Description |
| remember=5 | A password must be different from the last five passwords. |

Setting a Password Validity Period

- Best practices include setting a password validity period and prompting users to change passwords before a password expire.
- Modify the `/etc/login.defs` file to implement the setting.
- The `login.defs` file restricts accounts by setting items like the maximum password validity period and maximum password length (not apply to the root user). If the `/etc/shadow` file contains the same options, configurations in `/etc/shadow` take priority.
- When a user logs in to the system after the password expires, the system notifies the user of password expiration and prompts the user to change the password. If the user does not change the password, the user cannot log in to the system.

| Hardening Item | Description | Recommend Value | Hardened in openEuler |
|----------------|---|-----------------|-----------------------|
| PASS_MAX_DAYS | Maximum password validity period | 90 | No |
| PASS_MIN_DAYS | Minimum interval between password changes | 0 | Yes |
| PASS_WARN_AGE | Number of days in advance users are notified that their passwords are about to expire | 7 | Yes |

- To ensure system security, you are advised to set a password validity period and a prompt for users to change passwords before they expire.
- The `login.defs` file is used to set restrictions on user accounts, such as maximum password validity period and maximum length. Configurations in this file do not apply to the `root` user. If the `/etc/shadow` file contains the same options, settings in `/etc/shadow` take priority, that is, `/etc/shadow` has a higher priority than `/etc/login.defs`. When a user logs in to the system after the password expires, the system notifies the user that the password will soon expire, and prompts the user to change the password. If the user does not change the password, the user cannot log in to the system.

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 when passwords do not need to be recovered. The password encryption algorithm needs to be set to **sha512**, which is the default algorithm in openEuler. These settings can effectively prevent password disclosure and keep passwords secure.
- To set a password encryption algorithm, add the following configuration to **/etc/pam.d/password-auth** and **/etc/pam.d/system-auth**:

```
password sufficient pam_unix.so sha512 shadow nullok try_first_pass use_authtok
```

- **sha512** indicates that the SHA512 algorithm is used to encrypt passwords.

- 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.
- 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.

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.
- The configuration is as follows:

```
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
```

| Parameter | Description |
|-----------------|--|
| authfail | Captures user login failure events. |
| deny=3 | Locks an account after three consecutive login failures. |
| unlock_time=300 | Automatically unlocks an account after 300 seconds. |
| even_deny_root | Restricts the root user. |

Basic File Permissions

- File and directory security are essential to Linux, which treats everything as a file, and regards directory as a large file containing multiple files. Linux security is ensured by permissions and ownership.
- In openEuler, permissions and ownership for common directories, executable files, and configuration files are preset by default.
- Run the **chmod** and **chown** commands to modify the permission, owner, and owner group of a file.
 - Modify the file permission. For example, set the permission on the **/bin** directory to **755**.

```
chmod 755 /bin
```
 - Change the file ownership. For example, set the owner and group of the **/bin** directory to **root:root**.

```
chown root:root /bin
```

- File permission

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, openEuler has made related configurations by default. **PATH** is always initialized when **su** commands are used to switch users.
- Modify the **/etc/login.defs** file as follows:

```
ALWAYS_SET_PATH=yes
```

System Hardening Through File Permissions

- Common methods that harden the system by setting file permissions:
 - Set file permissions and ownership.
 - Delete unowned files.
 - Delete link files pointing to deleted files.
 - Add the sticky bit property for globally writable directories.
 - Disable global write on unauthorized files.
 - Restrict permissions to run **at** commands.
 - Restrict permissions to run **cron** commands.
 - Restrict permissions to run **sudo** commands.

File Permission Hardening – Setting File Permissions and Ownership

- File and directory security are essential to Linux, which treats everything as a file, and regards directory as a large file containing multiple files. Linux security is ensured by permissions and ownership.
- In openEuler, permissions and ownership for common directories, executable files, and configuration files are preset by default.
- Example: Set the permission on the **/bin** directory to **755**, and the owner and owner group to **root**.

```
chmod 755 /bin  
chown root:root /bin
```

File Permission Hardening – 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.

- Delete files whose user IDs do not exist.

```
find / -nouser  
rm -f filename
```

- Delete files whose group IDs do not exist.

```
find / -nogroup  
rm -f filename
```

- dirname* is the name of the directory to be searched. Pay special attention to key directories **/bin**, **/boot**, **/usr**, **/lib64**, **/lib**, and **/var**.
- After openEuler is installed, there may still be link files pointing to deleted files, causing these files to have corresponding functions. (Some of them are preconfigured and may be required by other components.) Handle these files based on the actual situation.
- follow:** If the **find** command finds a symbolic link file, it tracks the file to which the link points.

File Permission Hardening – Deleting Link Files Pointing to Deleted Files

- Link files pointing to deleted files may be exploited by malicious users. 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 required by other components.)

Perform the following operations based on the actual situation:

- Query link files pointing to deleted files.

```
find dirname -type l -follow 2>/dev/null
```

- Delete useless files.

```
find dirname -type l -follow 2>/dev/null
```

File Permission Hardening – 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 as follows:
 - Find globally writable directories.

```
find / -type d -perm -0002 ! -perm -1000 -ls | grep -v proc
```
 - Add the sticky bit property for globally writable directories. Replace **dirname** with the actual directory name.

```
chmod +t dirname
```

- Currently, the sticky bit **t** is valid only for directories. For directories with sticky bits, only the root user can delete all files, and common users can delete only the files created by themselves.
- SetUID=4,SetGID=2,t=1: the first digit indicates the special permission 7000.
- perm:** searches for files by file permission.

File Permission Hardening – Deleting Global Write on Unauthorized Files

- Globally writable files can be modified by any user in the system, which affects system integrity.
 - Display all globally writable files.

```
find / -type d ( -perm -o+w ) | grep -v proc  
find / -type f ( -perm -o+w ) | grep -v proc
```
 - 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
```
- Check whether a sticky bit is set for a file or directory. If the command output contains the **T** flag, the file or directory contains a sticky bit. Replace **filename** with the name of the file or directory to be queried.

```
ls -l filename
```

- Currently, the sticky bit **t** is valid only for directories. For directories with sticky bits, only the root user can delete all files, and common users can delete only the files created by themselves.
- SetUID=4,SetGID=2,t=1: the first digit indicates the special permission 7000.
- **-perm**: searches for files by file permission.

File Permission Hardening – Restricting Permissions to Run at Commands

- **at** commands are used to create tasks that are automatically executed at a specific time point. To prevent any user from arbitrarily using **at** commands and avoid system attacks, you need to specify the users who can use the commands as follows:

- Delete the **/etc/at.deny** file.

```
rm -f /etc/at.deny
```

- Change the owner of the **/etc/at.allow** file to **root:root**.

```
chown root:root /etc/at.allow
```

- Control the permission on the **/etc/at.allow** file. Only the **root** user can perform this operation.

```
chmod og-rwx /etc/at.allow
```

File Permission Hardening – Restricting Permissions to Run cron Commands

- **cron** commands are used to create routine tasks. To prevent any user from arbitrarily using **cron** commands and avoid system attacks, you need to specify the users who can use the commands as follows:

- Delete the **/etc/at.deny** file.

```
rm -f /etc/at.deny
```

- Change the owner of the **/etc/at.allow** file to **root:root**.

```
chown root:root /etc/at.allow
```

- Control the permission on the **/etc/cron.allow** file. Only the **root** user can perform this operation.

```
chmod og-rwx /etc/cron.allow
```

File Permission Hardening – 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 restrict **sudo** commands to only the root user. By default, openEuler does not restrict the permission of non-root users to use **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
```

Hardening the SSH Service

- The SSH protocol provides security assurance for remote login sessions and other network services, and effectively prevents information leakage during remote management. SSH encrypts transferred data to prevent domain name server (DNS) spoofing and IP spoofing. OpenSSH was created as an open source alternative to the proprietary SSH protocol.
- To harden the SSH service, set the algorithm and authentication parameters by modifying SSH service configurations when the system uses the OpenSSH protocol, thereby improving system security.
- Modify the `/etc/ssh/sshd_config` file to implement the setting.

- The SSH protocol provides security assurance for remote login sessions and other network services, and effectively prevents information leakage during remote management. SSH encrypts transferred data to prevent domain name server (DNS) spoofing and IP spoofing. OpenSSH was created as an open source alternative to the proprietary SSH protocol.

SSH Service Hardening Policy (1)

- Common hardening policies

| Hardening Item | Description | Recommended Value | Hardened in openEuler |
|----------------------|--|-------------------|-----------------------|
| MaxAuthTries | Sets the maximum number of authentication attempts. | 3 | No |
| PubkeyAuthentication | Sets whether public key authentication is allowed. | yes | Yes |
| PermitRootLogin | Sets whether to allow the root user to log in to the system using SSH. | no | No |
| PermitEmptyPasswords | Sets whether accounts with empty passwords can log in. | no | Yes |
| UsePAM | Sets whether to use PAM for login authentication. | yes | Yes |
| LoginGraceTime | The user must pass the authentication within the specified time limit. | 60 | No |
| SyslogFacility | Sets the log type of the SSH service, that is, authentication logs. | AUTH | Yes |
| LogLevel | Sets the level of SSHD logs. | VERBOSE | Yes |
| ClientAliveCountMax | Sets the maximum number of timeouts allowed. | 0 | Yes |
| AllowTcpForwarding | Sets whether to allow TCP forwarding. | no | Yes |

- **ClientAliveCountMax:** If the number of times that the client does not respond to the server reaches the specified value, the connection is automatically disconnected.
- **LoginGraceTime:** The value **0** indicates no limit. The default value is **120** seconds.

SSH Service Hardening Policy (2)

- Algorithm hardening policies

| Hardening Item | Description | Recommended Value | Hardened in openEuler |
|----------------|--|---|-----------------------|
| Ciphers | Sets the cryptographic algorithm for data transmission over SSH. | aes128-ctr,aes192-ctr,aes256-ctr, chacha20-poly1305@openssh.com, aes128-gcm@openssh.com, aes256-gcm@openssh.com | Yes |
| MACs | Sets the hash algorithm for SSH data verification. | hmac-sha2-512,hmac-sha2-512-etm@openssh.com, hmac-sha2-256,hmac-sha2-256-etm@openssh.com | Yes |
| KexAlgorithms | Sets the SSH key exchange algorithm. | curve25519-sha256, curve25519-sha256@libssh.org, diffie-hellman-group-exchange-sha256 | Yes |

- Other security suggestions

| Hardening Item | Description | Recommended Value | Hardened in openEuler |
|----------------|---|------------------------|-----------------------|
| ListenAddress | The SSH service only listens on specified IP addresses. | Specified IP addresses | No |

- ListenAddress:** For security purposes, you are advised to listen to only necessary IP addresses instead of 0.0.0.0 when using the SSH service.

Contents

1. OS Security Hardening

2. OS Network Security

- Netfilter Overview

- Iptables Description

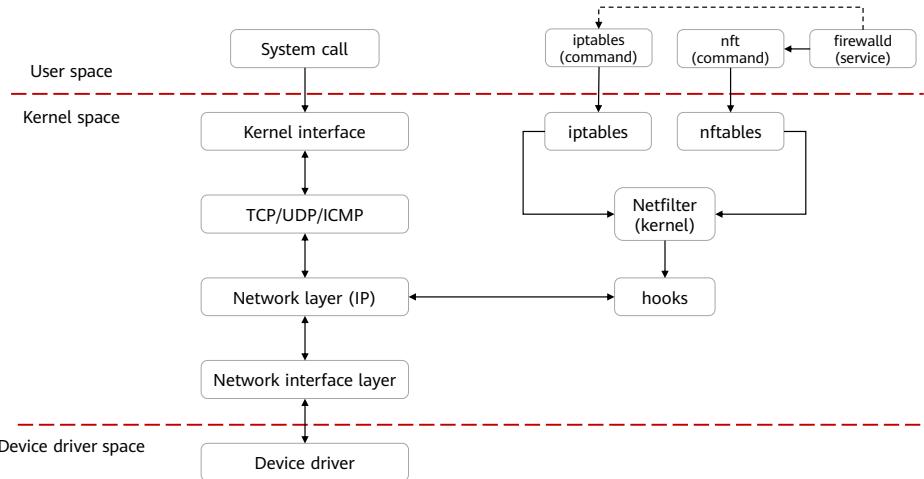
- Firewalld Description

3. SELinux Access Control

Overview of OS Network Security

- Network hardening is performed using the kernel-mode Netfilter, a data packet filtering framework. It matches data packets at the network layer according to preset rules and takes corresponding actions to control network traffic.
- Common user command line tools include iptables and firewalld.
- Based on zone rules and user-defined security policies, firewalld dynamically manages network connections and interfaces and formulates various firewall rules for services or applications.

Netfilter Ecosystem Tools



42 Huawei Confidential



- These hooks are triggered when each packet entering the network system passes through the protocol stack. A program can process network traffic on some critical paths by registering hook functions.
- **iptables:** a user command line tool for system administrators to manage Linux packet filtering rule sets.
- **nftables:** nft user command line tool based on specific network virtual devices and kernel framework for data packet classification. It reuses the existing Netfilter subsystem and implements more flexible, scalable, and high-performance packet classification. It is used to replace the current {IP, IP6, ARP, EB} tables.
- **firewalld:** Based on zone rules, firewalld dynamically manages security policies for network connections and interfaces, and formulates various instantaneous firewall rules for services or applications.
- Firewalld uses the iptables commands to directly configure iptables and uses NFS to manage firewall rules.

Netfilter Overview

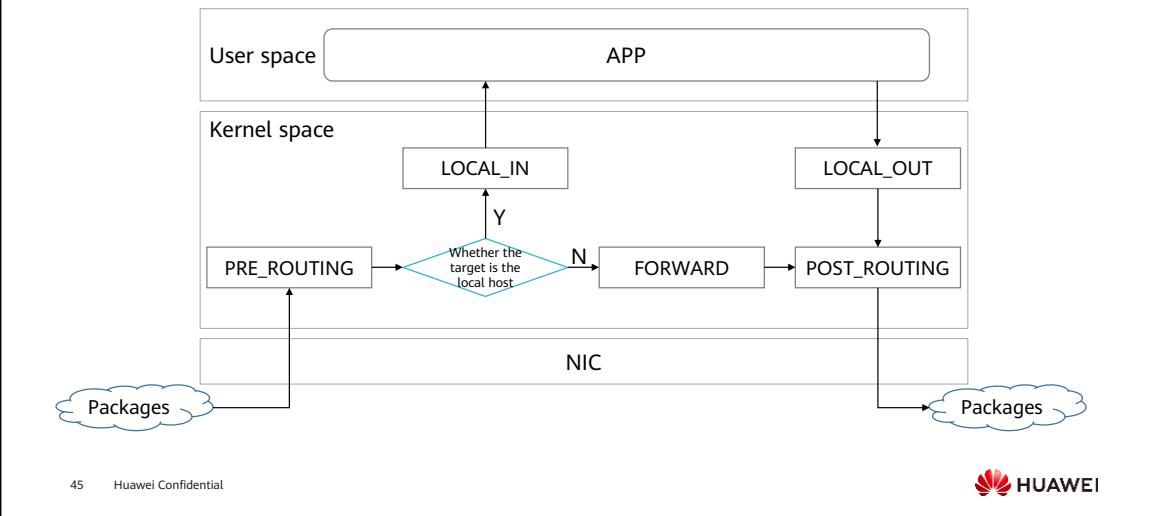
- Netfilter is a piece of data packet filtering software provided by the Linux kernel. It has the following features:
 - Stateless packet filtering
 - Stateful packet filtering
 - Network address and port translation
 - Flexible and extensible infrastructure
 - Multiple layers of APIs for third-party extensions

- Reference: <https://www.netfilter.org/index.html>

Netfilter Principles

- Netfilter uses five built-in kernel hooks to control network-layer data packets.
- Netfilter represents a set of hooks inside the Linux kernel, allowing specific kernel modules to register callback functions with the kernel's networking stack. These functions are called for every packet that traverses the respective hook within the networking stack.
 - **NF_IP_PRE_ROUTING:** This hook is triggered after a packet enters the network stack but before a routing decision is made.
 - **NF_IP_LOCAL_IN:** This hook is triggered when the route determines that a packet is sent to the local host.
 - **NF_IP_FORWARD:** This hook is triggered when the route determines that a packet needs to be forwarded to other hosts.
 - **NF_IP_LOCAL_OUT:** This hook is triggered before packets generated by the local process are sent to the network stack.
 - **NF_IP_POST_ROUTING:** This hook is triggered when a packet is about to be sent to the network after a routing decision is made.

Data Packet Filtering Process



- Packets sent to the local host:
 - PRE_ROUTING -> ROUTE -> LOCAL_IN
- Packets sent from the local host:
 - LOCAL_OUT -> POST_ROUTING
- Packets forwarded from the local host:
 - PRE_ROUTING -> ROUTE -> FORWARD -> POST_ROUTING

Contents

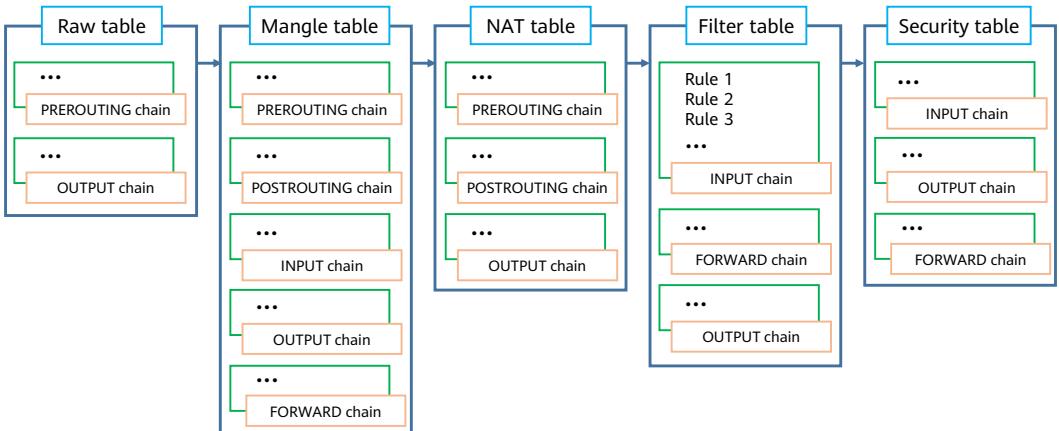
1. OS Security Hardening
2. **OS Network Security**
 - Netfilter Overview
 - **Iptables Description**
 - Firewalld Description
3. SELinux Access Control

Iptables Overview

- Iptables is widely used Linux firewall tool developed based on the framework Netfilter to enable system administrators to manage Linux packet filtering rule sets.

Tables, Chains, and Rules of Iptables

- Iptables uses tables to classify rules, and the internal rules of each table are chained.



48 Huawei Confidential



- This slide clearly shows the logical relationship between tables, chains, and rules.
- Vertically, each table contains chains, and each chain contains specific rules.
- A table specifies chain functions, and a chain provides specific operations to implement these functions, or rules.
- Horizontally, the same chain appears in different tables. Each chain also has auxiliary functions.
- To sum up, a chain is a set of rules, and a table is a set of rules with the same function. The serial connection of tables corresponds to processing procedures of different data packets.

Five Tables in Iptables

- A table is a set of rules with the same function. Each table is associated with a different kind of packet processing.
 - Filter table: the most widely used table in iptables. It is used to determine whether a data packet is sent to its destination address or deprecated.
 - Network address translation (NAT) table: changes the source or destination address of a data packet based on rules.
 - Mangle table: changes the header information of a data packet, such as TTL.
 - Raw table: provides iptables with a mechanism that bypasses connection tracking. For servers with heavy external service traffic, this table can avoid performance problems caused by connection tracking.
 - Security table: provides the function of adding SELinux features to data packets.

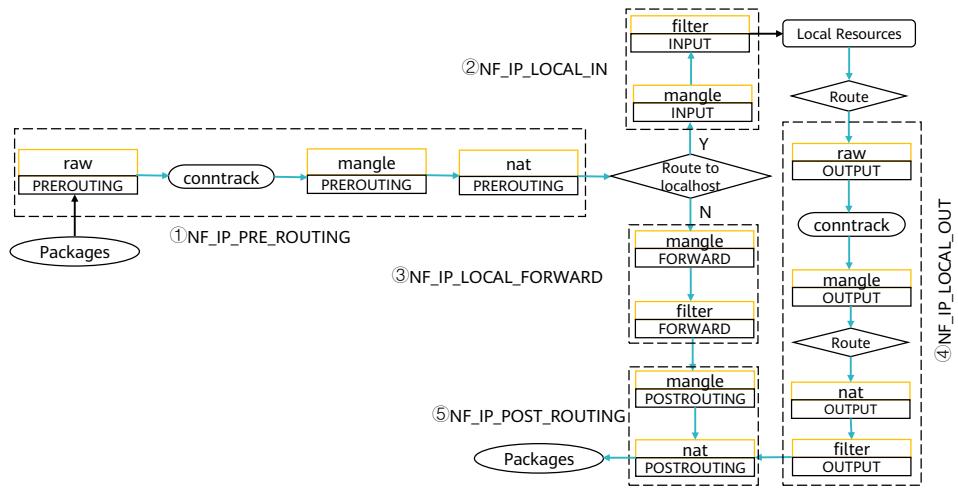
- Filter table: It is the most widely used table in iptables. It is used to filter data packets and determine whether a data packet is sent to its destination address or deprecated.
- NAT table: used for network address translation, including DNAT (mapping the local network to a public IP address for external services) and SNAT (used for LANs to share a public IP address, that is, serves as a gateway device).
- Mangle table: used to change the IP header information of a data packet. The available operations include ToS, TTL, and Mark.
- Raw table: only provides a framework for packets to bypass connection tracking.
- Security table: provides the function of adding SELinux features to data packets.
- The filter table is most frequently used, while the security table is not often used.

Five Chains in Iptables

- Chain: a set of rules applied in a specific order. Each packet passing through a chain needs to match all related rules for the corresponding action to be performed; otherwise, the default action is performed.
- The chain in iptables is implemented by Netfilter hooks. When data packets pass through the network stack, the kernel modules registered on these hooks are triggered in sequence.
 - **PREROUTING**: triggered by **NF_IP_PRE_ROUTING**
 - **INPUT**: triggered by **NF_IP_LOCAL_IN**
 - **FORWARD**: triggered by **NF_IP_FORWARD**
 - **OUTPUT**: triggered by **NF_IP_LOCAL_OUT**
 - **POSTROUTING**: triggered by **NF_IP_POST_ROUTING**

- Each packet passing through a chain matches the rules on the chain. If a packet meets the conditions of a rule, the action corresponding to the rule is performed. If no packet meets the conditions, the default action is performed.
- Think of the five chains in iptables as five toll stations on a highway. Only vehicles that meet the requirements of traffic flow are allowed to proceed, and those that do not meet the requirements are detained.

Working Principles of Iptables



Relationship of Tables and Chains in Iptables

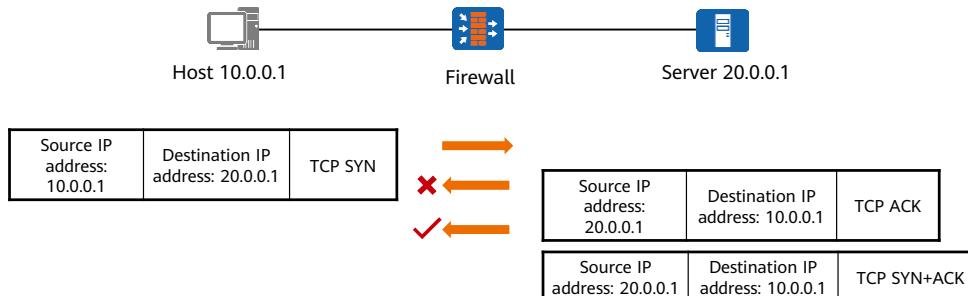
- The table shows the relationship between tables (vertical table headers) and chains (horizontal table headers). Y indicates that the table has a chain.

| Tables↓/Chains→ | PREROUTING | INPUT | FORWARD | OUTPUT | POSTROUTING |
|--------------------------|------------|-------|---------|--------|-------------|
| Route (routing decision) | | | | Y | |
| Raw | Y | | | Y | |
| Connection tracking | Y | | | Y | |
| Mangle | Y | Y | Y | Y | Y |
| NAT (DNAT) | Y | | | Y | |
| Route (routing decision) | Y | | | Y | |
| Filter | | Y | Y | Y | |
| Security | | Y | Y | Y | |
| NAT (SNAT) | | Y | | Y | Y |

- All status changes and computations are completed in the PREROUTING and OUTPUT chains in the NAT table. All connection traces are processed in the PREROUTING chain, excluding local packets which are processed by the OUTPUT chain
- The NAT table is divided into DNAT (modifying the destination address) and SNAT (modifying the source address) to display their priorities.
- The kernel determines the priority when registering the processing function so that the processing function can be called based on the priority when a hook is triggered. Multiple modules can be registered at the same hook, and the processing sequence is determined.

Connection Tracking Mechanism in Iptables

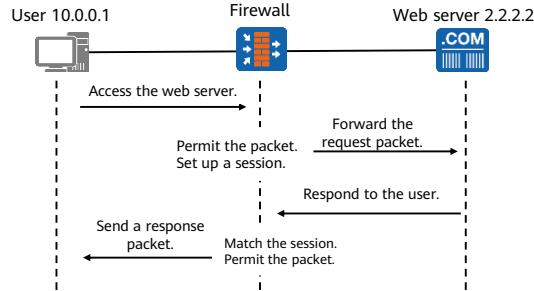
- In iptables, the firewall is stateful. The evaluation of each packet depends on the characteristics of a specific packet.
- The connection tracking feature established on Netfilter allows iptables to treat packets as part of a session rather than as independent and unrelated flows. After a packet reaches a network interface, it is judged by the connection tracking logic.



- Now, let's look at connection tracking of iptables.
- In iptables, the firewall is stateful. The evaluation of each packet is performed based on the characteristics of a specific packet.
- The connection tracking feature established on Netfilter allows iptables to treat packets as part of an existing connection or session rather than as independent and unrelated flows. After a packet arrives at a network interface, the packet is marked with a status flag.
- The connection tracking system maintains a memory data structure that records essential information about network connections. This includes details such as the source IP address, destination IP address, port numbers (for both TCP and UDP), protocol type, connection status, and timeout information. With this information, we can set filtering policies more flexibly.

Session Mechanism in Iptables

- The iptables treats all packets belonging to a connection as a data flow (session). A session is an entry used to record the connection status of protocols such as TCP, UDP, and ICMP, and is fundamental for iptables to forward packets.



| Session ID | Source IP Address | Source Port | Destination IP Address | Destination Port | Action |
|------------|-------------------|-------------|------------------------|------------------|--------------------|
| 1 | 10.0.0.1 | * | 2.2.2.2 | 80 | Permit the packet. |

54 Huawei Confidential



- The firewall uses the state-based packet control mechanism. The firewall determines the status of a connection by detecting only the first packet or a small number of packets. Subsequent packets are managed based on the connection status. This stateful detection mechanism rapidly improves the detection and forwarding efficiency of the firewall. The session table records the status of a connection. When forwarding TCP, UDP, and ICMP packets, the device needs to query the session table for the corresponding connection and take subsequent measures.

Iptables Syntax Rules

- On a chain, a rule is used to block or permit different connections and packets. Each row inserted into a chain is a rule.
- From the perspective of the kernel, a rule is a statement that determines how to process a package. If a package meets all conditions (that is, complies with the match statement), run the **target** or **jump** command.
- Syntax format:
 - **iptables [-t table] command [match] [target/jump]**
 - **[-t table]**: specifies the table to be operated. By default, the table is a filter table.
 - **[COMMAND]**: specifies the operation to be performed by iptables on the submitted rule.
 - **[match]**: specifies a matching rule.
 - **[target/jump]**: determines where the packets that meet the conditions are sent.

Iptables Syntax Rule – COMMAND

- The following commands are available for iptables. By default, operations are performed on the filter table. **-h** can be used to provide brief syntax description.

| Command | Description |
|---------------------------|--|
| -A,--append | Adds a rule at the end of a specified chain. |
| -D,--delete | Deletes a rule from a specified chain. |
| -R,--replace | Replaces a rule in a specified line of the selected chain. |
| -I,--insert | Inserts a rule with a specified sequence number into the selected chain. |
| -L,--list | Displays all rules of the selected chain. |
| -F,--flush | Flushes rules of a specified chain. |
| -Z,--zero | Zeros the counter of the selected chain. |
| -N,--new-chain | Adds a user-defined chain. |
| -X, --delete-chain | Deletes a user-defined chain. |
| -P, --policy | Specifies the default policy (DROP or ACCEPT) of a chain. |
| -E, --rename-chain | Renames a user-defined chain. |

- P:** There are only two default policies: **ACCEPT** and **DROP**.
- X:** Rules of a user-defined chain must be cleared before deleting that chain.
- Z:** There are two counters in a chain, which record the number of matched packets and bytes.

Iptables Syntax Rule – COMMAND

- With related options, more functions can be implemented.

| Option | Command | Description |
|--------------------|---|--|
| -v, --verbose | --list, --append, --insert, --delete, --replace | Provides detailed information about related commands. |
| -x, --exact | --list | Displays the accurate value of the counter. |
| -n, --numeric | --list | Displays the IP address and port number as numbers. |
| --line-numbers | --list | Lists the sequence number of each rule in a chain. |
| -c, --set-counters | --insert, --append, --replace | Sets the counter value when a rule is created or changed. |
| --modprobe | All | Enables iptables to detect and load the module to be used. |

Iptables Syntax Rule – MATCH

- MATCH: describes the detailed characteristics of a packet and the actions performed based on matching rules. Options include generic matching and extended matching.
- Generic matching applies to all rules and can be used directly.
- Extended matching includes explicit matching and implicit matching.

| Generic Matching Option | Description | Example |
|--------------------------|---|----------------------------------|
| -p, --protocol | Matches a specified protocol. | iptables -A INPUT -p tcp |
| -s, --src, --source | Matches packets based on the source IP address. | iptables -A INPUT -s 192.168.1.1 |
| -d, --dst, --destination | Matches packets based on the destination IP address. | iptables -A INPUT -d 192.168.1.1 |
| -i, --in-interface | Matches packets with the inbound local network interface. | iptables -A INPUT -i eth0 |
| -o, --out-interface | Matches packets with the outbound local network interface. | iptables -A FORWARD -o eth0 |
| -f, --fragment | Matches the second fragment or subsequent parts of a fragmented packet. | iptables -A INPUT -f |

Iptables Syntax Rule – MATCH

- Implicit matching operations are automatically or implicitly loaded into the kernel. Currently, there are three implicit matching types (TCP matches, UDP matches, and ICMP matches), each of which applies only to the corresponding protocol.

| TCP Matches | Description | Example |
|-----------------------------|---|---|
| --sport, --source-port | Matches packets based on the source port of TCP packets. | iptables -A INPUT -p tcp --sport 22 |
| --dport, --destination-port | Matches packets based on the destination port of TCP packets. | iptables -A INPUT -p tcp --dport 22 |
| --tcp-flags | Matches specified TCP flags. | iptables -p tcp --tcp-flags SYN,FIN,ACK SYN |
| UDP Matches | Description | Example |
| --sport, --source-port | Matches packets based on the source port of UDP packets. | iptables -A INPUT -p udp --sport 53 |
| --dport, --destination-port | Matches packets based on the destination port of UDP packets. | iptables -A INPUT -p udp --dport 53 |
| ICMP Matches | Description | Example |
| --icmp-type | Matches packets based on the ICMP type. | iptables -A INPUT -p icmp --icmp-type 8 |

Iptables Syntax Rule – MATCH

- Explicit matching must be loaded using **-m** or **--match**. Currently, the available modules include Limit, MAC, Multiport, Owner, State, and TTL. The following lists some common options.

| Option | Description | Example |
|--------------------|--|---|
| --limit | Sets the maximum average matching rate. | iptables -A INPUT -m limit --limit 3/hour |
| --limit-burst | Sets the maximum number of matched packets per unit time. | iptables -A INPUT -m limit --limit-burst 5 |
| --mac-source | Matches packets based on the source MAC address of the packets. | iptables -A INPUT -m mac --mac-source 00:00:00:00:00:00 |
| --source-port | Matches multiple source ports. | iptables -A INPUT -p tcp -m multiport --source-port 22,53,80 |
| --destination-port | Matches multiple destination ports. | iptables -A INPUT -p tcp -m multiport --destination-port 22,53,80 |
| --uid-owner | Matches output packets based on the user ID of the generated packet. | iptables -A OUTPUT -m owner --uid-owner 500 |
| --gid-owner | Matches output packets based on the user group ID of the generated packet. | iptables -A OUTPUT -m owner --gid-owner 0 |
| --state | Specifies the status of the packet to be matched. | iptables -A INPUT -m state --state RELATED,ESTABLISHED |
| --ttl | Matches packets based on the TTL value. | iptables -A OUTPUT -m ttl --ttl 60 |

Iptables Syntax Rules – Targets/Jumps

- **target** or **jump** determines where the package that meets the conditions is sent. The syntax is **--jump target** or **-j target**.
 - **ACCEPT/DROP target**: permits or discards the packets that meet the conditions.
 - **REJECT target**: rejects the packets that meet the conditions and returns error information to the sender.
 - **DNAT target**: translates the destination network address. When a packet is matched, it is routed to the correct host and network.
 - **SNAT target**: translates the source network address, which changes the source addresses of all packets sent from the local network to the IP address of the Internet connection.
 - **MASQUERADE target**: translates the source network address, without specifying the source address. The IP address of the network interface is automatically obtained.
 - **REDIRECT target**: forwards internal packets of the host to another port, and all locally generated packets are mapped to 127.0.0.1.
 - **LOG target**: records package information, which is generally implemented by the kernel log tool syslogd.
 - **RETURN target**: returns a packet to the upper layer. The sequence is child chain > parent chain > default policy.

Saving and Restoring Rule Sets of Iptables

- The iptables provides two useful tools to process large rule sets. They store rules in a file in a special format or restore rules from this file.
 - **iptables-save [-c] [-t table]**: saves the current rule to a file.
 - **-c**: saves the values of the packet and byte counters. If the firewall is restarted, the counting program will not be interrupted. This parameter is not used by default.
 - **-t**: specifies the table to be saved. By default, all tables are saved.
 - **iptables-restore [-c] [-n]**: loads the rule set saved by **iptables-save**.
 - **-c, --counters**: loads the packet and byte counters.
 - **-n,--noflush**: does not flush existing tables or rules in the table. By default, all existing rules are cleared.

- **iptables-save** and **iptables-restore** can significantly improve the speed of loading and saving rules.
- These two tools are ideal for processing a huge set of rules.

Iptables Configuration Example – Access Control

- Allow IP addresses from all sources to access TCP port 22 of the local host.

```
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
```

- 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
```

- Deny the IP address from 104.224.147.43 to access TCP port 80.

```
iptables -A INPUT -p tcp -s 104.224.147.43 --dport 80 -j REJECT
```

- Save iptables rules.

```
iptables-save > /etc/sysconfig/iptables
```

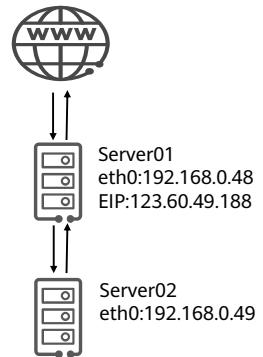
Iptables Configuration Example – NAT

- SNAT: Server02 accesses the Internet through Server01. The source IP address displayed on the Internet is the external IP address of Server01.

```
# Run the following commands on Server01:  
sysctl -w net.ipv4.ip_forward=1  
iptables -P INPUT ACCEPT  
iptables -P OUTPUT ACCEPT  
iptables -P FORWARD ACCEPT  
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 192.168.0.48
```

- DNAT: External users directly access services provided by Server02 without an external IP address.

```
# Run the following commands on Server01:  
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  
iptables -t nat -A POSTROUTING -d 192.168.0.48 -p tcp -m tcp --dport 80 -j SNAT --to-source  
192.168.0.49
```



Contents

1. OS Security Hardening

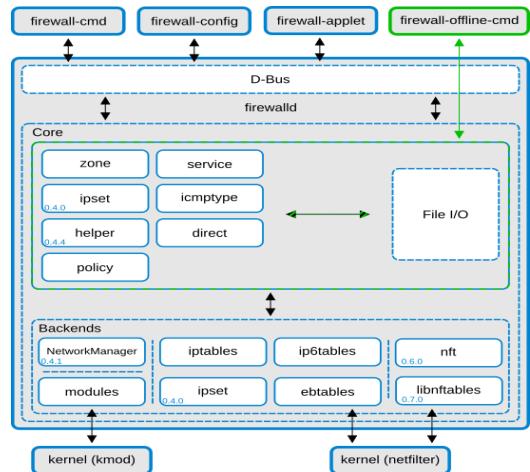
2. OS Network Security

- Netfilter Overview
- Iptables Description
- **Firewalld Description**

3. SELinux Access Control

Firewalld Architecture

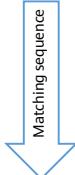
- Firewalld has a two-layer design comprising D-Bus layer on top of the core layer .
 - The core layer processes various configuration files (XML) and backend components, such as iptables, ip6tables, ebttables, and modules.
 - The D-Bus interface is the main method of changing and creating firewall configurations. This interface is used by all tools provided by firewalld, such as firewall-cmd, firewall-config, firewall-applet, and firewall-offline-cmd.



- `firewall-offline-cmd` does not communicate with firewalld. Instead, it directly uses the firewalld core and I/O handlers to change and create the firewalld configuration file. `firewall-offline-cmd` can be used when firewalld is running, but this is not recommended, because it can only alter the permanent configuration that is visible in firewall after about five seconds.
- Image source: <https://firewalld.org/documentation/architecture.html>

Firewalld Policies

- Primary functions include controlling access to and from the network, safeguarding the specific network against untrusted networks, and ensuring legal communication between two networks. Firewalld implements these functions mainly through security policies.
- Its basic design principle is to deny all traffic by default unless otherwise specified. This ensures that firewalld can protect the cyber security once it is networked.
- The security policy matching process is as follows:



| Policy ID | Matching Condition | | | | Action |
|-----------|----------------------|----------------------|-----|----------------------|-------------|
| Policy 1: | Matching condition 1 | Matching condition 2 | ... | Matching condition N | Permit/Deny |
| Policy 2: | Matching condition 1 | Matching condition 2 | ... | Matching condition N | Permit/Deny |
| ... | | | | | |
| Policy N: | Matching condition 1 | Matching condition 2 | ... | Matching condition N | Permit/Deny |
| Default: | Any | | | | Deny |

- Firewalld uses policies to control access to and from the network. By default, it denies all traffic.
- The return path of traffic is allowed by default due to the connection tracking mechanism of the firewall.

Firewalld Zones

- In firewalld, a zone determines the trust level for a connection, interface, or source address. It follows a one-to-many relationship: a connection, interface, or source address can belong to only one zone, but a zone can be associated with multiple connections, interfaces, or source addresses.
- Predefined zones:

| Zone | Description |
|----------|---|
| trusted | In a trusted zone, all traffic is permitted, which is equivalent to disabling the firewall function. |
| home | In this zone, incoming return packets that initiate connections are permitted. By default, services such as SSH, MDNS, and ipp-client are permitted. |
| internal | Its function is the same as that of the home zone. |
| work | In this zone, incoming return packets that initiate connections are permitted. By default, services such as SSH and dhcp6-client are permitted. |
| external | In this zone, incoming return packets that initiate connections are permitted. By default, the SSH service is permitted, and address spoofing is enabled. |
| dmz | In this zone, incoming return packets that initiate connections are permitted. By default, the SSH service is permitted. |
| block | In this zone, incoming return packets that initiate connections are permitted. |
| drop | All data packets entering this zone are discarded, and no packet is returned. Incoming return packets that initiate connections are permitted. |

Other Important Notes of Firewalld

- Supported configuration methods:
 - WebUI
 - firewall-config GUI
 - firewall-cmd
 - XML file
- Configuration files:
 - **/usr/lib/firewalld**: default firewall configuration file. You are advised not to modify it.
 - **/etc/firewalld**: user-defined configuration file. If there is no configuration file in this directory, the firewall uses the default configuration.
- Firewalld states:
 - Runtime: A rule takes effect immediately after being modified, but is not saved to the permanent configuration. And the rule becomes invalid after firewalld is restarted. It is recommended that the rule be used for rule testing.
 - Permanent: After the configuration is complete, you need to reload firewalld for the configuration to take effect. The permanent configuration is stored in the configuration file.

Firewalld Configuration Method – firewall-cmd

- Syntax: **firewall-cmd [option...]**

| Common Option | Description |
|------------------------|---|
| -h,--help | Obtains command help information. |
| -V,--version | Obtains the firewall version. |
| -q,--quiet | Does not output status information. |
| Status Option | Description |
| --state | Obtains the firewall status. |
| --reload | Reloads firewall rules and retains active connections. |
| --complete-reload | Reloads the firewall and interrupts active connections. |
| --runtime-to-permanent | Saves the current configuration as a permanent configuration. |
| --check-config | Checks whether the configuration is correct. |
| Log Option | Description |
| --get-log-denied | Obtains log information. |
| --set-log-denied=value | Sets log recording rules. |

Firewalld Configuration Method – firewall-cmd

```
[root@firewalld ~]# firewall-cmd --get-default-zone
public
[root@firewalld ~]# firewall-cmd --set-default-zone=home
success
[root@firewalld ~]# firewall-cmd --get-default-zone
home
[root@firewalld ~]# firewall-cmd --get-active-zones
home
    interfaces: ens3
[root@firewalld ~]# firewall-cmd --get-services
RH-Satellite-6 RH-Satellite-6-capsule amanda-client amanda-k5-
client amqp amqps.....
[root@firewalld ~]# firewall-cmd --get-zone-of-interface=ens3
home
[root@firewalld ~]# firewall-cmd --get-zone-of-source=10.0.0.101
no zone
[root@firewalld ~]# firewall-cmd --info-zone=home
home (active)
    target: default
    icmp-block-inversion: no
    interfaces: ens3
.......
```

| Option | Description |
|-----------------------------------|--|
| --get-default-zone | Obtains the default zone. |
| --set-default-zone=zone | Sets the default zone. |
| --get-active-zones | Obtains details about the current active zone. |
| --get-zones | Obtains predefined zones. |
| --get-services | Obtains the service. |
| --get-icmptypes | Obtains the ICMP type. |
| --get-zone-of-interface=interface | Obtains the zone bound to an interface. |
| --get-zone-of-source=source | Obtains the zone bound to the source address. |
| --info-zone=zone | Obtains information about a specified zone. |
| --list-all-zones | Lists information about all zones. |
| --new-zone=zone | Adds a new zone. |
| --delete-zone=zone | Deletes the current zone. |

Firewalld Configuration Method – firewall-cmd

```
[root@firewalld ~]# firewall-cmd --get-policies  
allow-host-ipv6  
[root@firewalld ~]# firewall-cmd --info-policy=allow-host-ipv6  
allow-host-ipv6 (active)  
priority: -15000  
target: CONTINUE  
ingress-zones: ANY  
egress-zones: HOST  
services:  
ports:  
protocols:  
masquerade: no  
forward-ports:  
source-ports:  
icmp-blocks:  
rich rules:  
    rule family="ipv6" icmp-type name="neighbour-advertisement" accept  
    rule family="ipv6" icmp-type name="neighbour-solicitation" accept  
    rule family="ipv6" icmp-type name="router-advertisement" accept  
    rule family="ipv6" icmp-type name="redirect" accept
```

| Option | Description |
|---------------------------------|---|
| --get-policies | Obtains predefined policies. |
| --info-policy=policy | Obtains information about a specified policy. |
| --list-all-policies | Obtains all policies. |
| --new-policy=policy | Adds a policy. |
| --new-policy-from-file=filename | Adds a policy through a file. |
| --path-policy=policy | Obtains the policy configuration file path. |
| --delete-policy=policy | Deletes a policy. |
| --load-policy-defaults=policy | Loads the default policy value. |

Firewalld Configuration Method – firewall-cmd

```
[root@firewalld ~]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens3
  .....
[root@firewalld ~]# firewall-cmd --change-interface=ens3 --
zone=home
success
[root@firewalld ~]# firewall-cmd --add-service=http --zone=public
success
[root@firewalld ~]# firewall-cmd --list-all | grep http
  services: dhcpcv6-client http mdns ssh
[root@firewalld ~]# firewall-cmd --add-port=8080/tcp --
zone=public
success
[root@firewalld ~]# firewall-cmd --list-all | grep 8080
  ports: 8080/tcp
[root@firewalld ~]# firewall-cmd --add-port=8080/tcp --
zone=public --permanent
success
[root@firewalld ~]# firewall-cmd --reload
success
```

| Option | Description |
|-----------------------------|---|
| --list-all | Lists details about the active zone. |
| --zone=zone | Specifies the zone to be configured or viewed. |
| --add-interface=nic | Diverts traffic from the NIC to a specified zone. |
| --change-interface=nic | Associates an NIC with a zone. |
| --get-services | Displays preset services. |
| --add-service=service | Sets the service traffic allowed in the default zone. |
| --add-port=port/protocol | Sets the port traffic allowed in the default zone. |
| --remove-service=service | Sets the service traffic not allowed in the default zone. |
| --remove-port=port/protocol | Sets the port traffic not allowed in the default zone. |
| --permanent | Configures a permanent policy. |
| --reload | Reloads a policy. |

Firewalld Configuration Method – Modifying the XML File

- The XML file containing firewalld rules are stored in the **/etc/firewalld/zones** directory. By default, the directory is empty. If a rule is changed to a permanent one, its corresponding file is automatically generated.
- After you perform operations on other zones, the corresponding XML file is automatically generated.
- After the XML file is modified, you need to run the **--reload** command on the firewalld service for the modification to take effect.

```
[root@firewalld zones]# ls /etc/firewalld/zones
public.xml
[root@firewalld zones]# cat public.xml
<?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="dhcpcv6-client"/>
<port port="8080" protocol="tcp"/>
<forward/>
</zone>
```

Configuring Firewalld Rich Rules

- By using rich rules of firewalld, you can define more complex and refined rules.
- Example 1: Accept all traffic from host 192.168.1.100.

```
[root@firewalld ~]# firewall-cmd --add-rich-rule="rule family="ipv4" source address="192.168.1.100" accept"
success
```
- Example 2: Allow host 192.168.1.100 to ping the local host.

```
[root@firewalld ~]# firewall-cmd --add-rich-rule="rule family="ipv4" source address="192.168.1.100" protocol value="icmp" accept"
success
```
- Example 3: Allow host 192.168.1.100 to perform SSH on the local host.

```
[root@firewalld ~]# firewall-cmd --add-rich-rule="rule family="ipv4" source address="192.168.1.100" port protocol="tcp" port=22 accept"
success
```
- Example 4: Forward the traffic from port 80 of host 192.168.1.100 to port 8080 of host 192.168.1.101.

```
[root@firewalld ~]# firewall-cmd --add-rich-rule="rule family="ipv4" source address="192.168.1.100" forward-port port=80
protocol=tcp to-port=8080 to-addr=192.168.1.101"
```
- Example 5: Perform NAT forwarding on host 192.168.1.0/24.

```
[root@firewalld ~]# firewall-cmd --add-rich-rule="rule family="ipv4" source address="192.168.1.0/24" masquerade"
success
```

Contents

1. OS Security Hardening
2. OS Network Security
- 3. Mandatory Access Control on SELinux**

Introduction to SELinux

- Discretionary access control (DAC) determines resource access by users, groups, and other permissions. It does not allow the system administrator to create comprehensive and fine-grained security policies. SELinux (Security-Enhanced Linux) is a module of the Linux kernel and a security subsystem of Linux that implements mandatory access control (MAC). Each process and system resource has a special security label. In addition to the principles specified by the DAC, the SELinux determines whether each type of process has the permission to access a type of resource.
- By default, openEuler uses SELinux to improve system security. SELinux has three modes:
 - **Permissive**: outputs alarms but does not forcibly execute the security policies.
 - **Enforcing**: security policies are forcibly executed.
 - **Disabled**: security policies are not loaded.

Viewing and Converting the SELinux Mode

- Run **getenforce** to query the current SELinux mode.

```
[root@firewalld ~]# getenforce  
Enforcing
```

- Run **setenforce** to temporarily change the SELinux mode. The value **0** indicates the permissive mode and the value **1** indicates that the enforce mode.

```
[root@first-day-01 ~]# setenforce 0  
[root@first-day-01 ~]# getenforce  
Permissive  
[root@first-day-01 ~]# setenforce 1  
[root@first-day-01 ~]# getenforce  
Enforcing
```

- Modify the **/etc/selinux/config** file to permanently change the SELinux mode, and restart the system for the modification to take effect.

```
# 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
```

Security Context of SELinux

- In SELinux, all files, ports, and processes have a unique security label called the SELinux context.
- A security context consists of five elements in the format "**user:role:type:sensitivity:category**".
 - **user**: indicates the type of the user who logs in to the system.
 - **role**: defines the usage of files, ports, and processes.
 - **type**: defines data types.
 - **sensitivity**: defines the security level.
 - **category**: distinguishes specific organizations.
- When the SELinux policy is targeted, you only need to pay attention to **type**.

Checking SELinux Security Contexts

- SELinux security contexts can be classified into actual contexts and default contexts. Actual contexts are not stored in the file system, and default contexts are stored in the binary SELinux policy library.
- Use the **-Z** option to check actual contexts.
 - Check actual security contexts of a process.

```
[root@first-day-01 ~]# ps axZ
system_u:system_r:init_t:s0  1 ?      Ss  0:01 /usr/lib/systemd/systemd --switched-root -system.....
```
 - Check actual security contexts of a file.

```
[root@first-day-01 ~]# ls -Z /etc/ssh
system_u:object_r:etc_t:s0 moduli
```
 - Check actual security contexts of a port.

```
[root@first-day-01 ~]# ss -lnpZ | grep 22
tcp  LISTEN 0 ..... users:(("sshd",pid=661,proc_ctx=system_u:system_r:sshd_t:s0-s0:c0.c1023,fd=3))
```
- Run the **semanage** command to check default contexts.

```
[root@first-day-01 ~]# semanage fcontext -l | grep www
/home/[^/]+/(www|web|public_html)(/.*)?/\.htaccess regular file    unconfined_u:object_r:httpd_user_htaccess_t:s0
```

- Actual contexts do not become invalid after the system is restarted. They become invalid only when the system relabels all files.

Managing Security Contexts of Files in SELinux

- The **chcon** command is used to modify the actual security context of a file. The command syntax is **chcon -t *TYPE* *file/directory***, where **TYPE** indicates the specific context, **file** indicates the specified file, and **directory** indicates the specified directory.
- The **semanage fcontext** command is used to modify the default security context of a file. The command syntax is **semanage fcontext -option *file/directory***. The common options are as follows:
 - **-a**: adds a security context.
 - **-d**: deletes a security context.
 - **-t**: specifies the context type of a file.
 - **-l**: lists security contexts of a file.
- The **restorecon** command is used to restore the default settings. The command syntax is **restorecon -option *file/directory***. The common options are as follows:
 - **-v**: views detailed information.
 - **-R**: recursively modifies a directory.

Managing Security Contexts of Ports in SELinux

- SELinux can control network traffic by labeling network ports. When a process wants to listen on a port, SELinux checks whether the process-related label can be bound to the port label.
- If a service runs on a non-standard port, SELinux intercepts the service. In this case, you need to modify the security context of the port to match the service process.
- You can run command **semanage port -option port** to change the security context of a port. The common options are as follows:
 - **-a**: adds a security context.
 - **-d**: deletes a security context.
 - **-l, --list**: lists records of the port object type.
 - **-t TYPE, --type** : lists the SELinux type for an object.

Boolean Value of SELinux

- The SELinux Boolean value is used to adjust an SELinux policy by enabling or disabling rules.
- If the value is set to **on**, the corresponding SELinux rule is enabled. If the value is set to **off**, the corresponding SELinux rule is disabled.
- Commands related to the SELinux Boolean value:
 - **semanage boolean -l**: checks all SELinux Boolean values.
 - **getsebool -a**: checks all valid Boolean values.
 - **setsebool -P rule on/off**: enables or disables a Boolean value. Replace **rule** with the actual rule.

SELinux Log Service

- By default, SELinux uses the auditd service to record rejection information.
 - Log path: **/var/log/auditd/audit.log**
- Incorrect file types are often to blame for SELinux access denials. The following takes the rejection message of the Apache HTTP server as an example:

```
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
```

 - **avc: denied { getattr }**: permission denied. The source process is trying to read the status information of the target file.
 - **pid=1958 comm="httpd"**: process name and ID.
 - **path="/mywebsite/index.html"**: object that the process attempts to access.
 - **scontext=system_u:system_r:httpd_t:s0**: process context that rejects the operation.
 - **tcontext=unconfined_u:object_r:default_t:s0**: context of the object accessed by the process.
 - **permissive=0**: current SELinux mode.

Quiz

1. (Single Choice) Which of the following methods allows firewalls and some routers to direct addresses to untrusted networks by hiding network addresses?
 - A. Address filtering
 - B. NAT
 - C. Inversion
 - D. IP spoofing
2. (True or False) Generally, to ensure system security, common applications do not need to be operated by the root user.

- B
- True

Summary

- This course covers fundamentals and hardening strategies for OS security, the basics of Netfilter, iptables and firewalld that are built on the Netfilter framework, and SELinux mandatory access control.

Thank you.

把数字世界带入每个人、每个家庭、

每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and organization for a fully connected, intelligent world.

Copyright©2024 Huawei Technologies Co., Ltd.
All Rights Reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.



openEuler System Monitoring



Foreword

- System metrics give a broad overview of OS and application statuses. This course describes the basic system parameters, and briefly introduces Zabbix, and a mainstream system tool.

Objectives

- On completion of this course, you will understand:
 - Linux commands for the system status.
 - Installation, deployment, and basic operations of Zabbix.

Contents

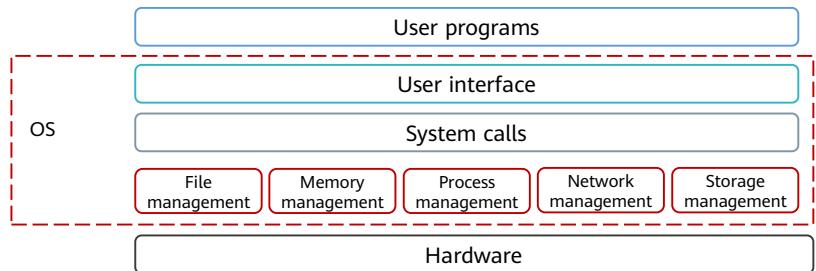
- 1. Commands for Checking openEuler Performance**
2. System Monitoring with Zabbix

Operating System Overview

- The operating system (OS) acts as an environment between the user and computer hardware, on which programs are executed.
- Computer hardware are the base computing resources for the system, and includes the central processing unit (CPU), memory, and input/output (I/O) devices. Applications determine how to use resources and solve computational problems, whereas the OS manages hardware resources and their usage across user applications.
- To use the OS more efficiently, users need to know the utilization of hardware resources such as the CPU, memory, and I/O.

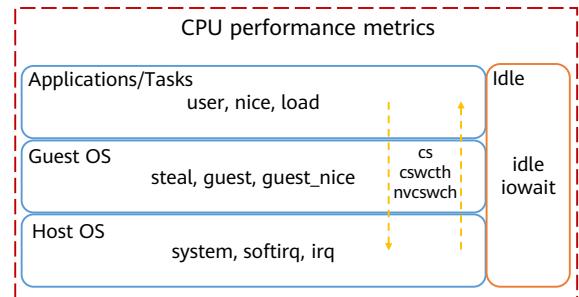
OS Performance Metrics

- CPU
- Memory
- I/O
 - Drive I/O
 - Network I/O



Commands for Checking the CPU Status

- CPU usage
 - Common tools: **mpstat**, **top**, and **ps**
- Context switch
 - Common tools: **pidstat** and **vmstat**
- Average load
 - Common tools: **uptime** and **top**



Key Memory Metrics and Monitoring Tools

- System memory
 - Physical memory
 - Swap space memory
 - Buffer
 - Cache
- Process memory
 - Virtual memory (VIRT)
 - Resident memory (RES)
 - Shared memory (SHR)
- Common tool: **top**
- Common tools: **free** and **vmstat**

Key Drive I/O Metrics and Monitoring Tools

- Input/output operations per second (IOPS): number of read and write requests that a drive can process in one second
- Throughput: drive read/write traffic every second
- Common tools: **iostat** and **sar**

Network I/O Metrics and Monitoring Tools

- Network connection status: TCP, UDP, HTTP, etc.
- Response time: total time it takes for a user to receive a response after sending a request
- Bandwidth: amount of data that can be transmitted per unit time
- Common tools: **netstat**, **ping**, **sar**, and **iperf3**

Checking CPU Usage Statistics – mpstat

- The **mpstat** command reports CPU statistics.
- Syntax:
 - `mpstat [-A] [--dec={ 0 | 1 | 2 }] [-n] [-u] [-T] [-V] [-I { keyword[...] | ALL }] [-N { node_list | ALL }] [-o JSON] [-P { cpu_list | ALL }] [interval [count]]`
- Common options of the **mpstat** command:
 - **-P**: displays the statistics of specified CPUs. For example, **-P ALL** displays the statistics of all CPUs, and **-P 1,2** displays the statistics of the second and third CPUs. You can add the display interval after the specified CPUs. For example, **-P 1,2 2** indicates that the output is refreshed every two seconds.
 - **-T**: displays topology elements. It is usually used together with **-u**.
 - **-u**: specifies the information to display.
 - **-o**: displays statistics in JSON format.

Checking CPU Usage Statistics – mpstat

- CPU status displayed in the **mpstat** output:
 - **CPU**: processor ID. **all** indicates the average value of all processors.
 - **%usr**: percentage of CPU time used by user-mode processes
 - **%nice**: percentage of CPU time used by low-priority user-mode processes (whose nice value is greater than 0)
 - **%sys**: percentage of CPU time used by system (kernel-mode) processes
 - **%soft**: percentage of CPU time used to handle software interrupts
 - **%irq**: percentage of CPU time used to handle hardware interrupts
 - **%steal**: percentage of time that a VM spends waiting involuntarily
 - **%guest**: percentage of time that the CPU spends running a virtual processor
 - **%gnice**: percentage of CPU time used by low-priority VMs (similar to **%gnice**)
 - **%idle**: percentage of idle CPU time
 - **%iowait**: percentage of the time that the CPU spends waiting for I/O operations

- CPU resources used by user-mode processes with a nice value greater than 0 are reported under **%nice**. CPU resources used by other user-mode processes are reported under **%usr**.

Viewing Process Context Switches – pidstat

- The **pidstat** command displays all processes managed by the current kernel.

```
[root@openEuler ~]# pidstat -w 1 2
Linux 5.10.0-60.18.0.50.oe2203.x86_64 (openEuler)      05/17/2023      _x86_64_      (2 CPU)

12:29:13 PM   UID      PID  cswch/s nvcswch/s  Command
12:29:14 PM   0       12    1.98    0.00  rCU_sched
12:29:14 PM   0       30    1.98    0.00  kcompactd0
12:29:14 PM   0      1171    1.98    0.00  kworker/1:0-mm_percpu_wq
12:29:14 PM   0      1402    1.98    0.00  kworker/0:2-events
12:29:14 PM   0      1406    0.99    0.99  pidstat
```

- cswch/s:** total number of voluntary context switches the task made per second
 - A voluntary context switch occurs when a process is blocked due to unavailable resources.
- nvcswch/s:** total number of non voluntary context switches the task made per second
 - An involuntary context switch occurs when the CPU time slice of a process expires, forcing the process to relinquish the processor.

Viewing Average CPU Load – uptime

- The **uptime** command displays the system time and average load.

```
[root@openEuler ~]# uptime  
15:00:30 up 36 min, 2 users, load average: 0.75, 0.59, 0.25
```

- **36 min:** system running duration
- **2 users:** number of users currently logged in
- **load average:** average system load over the last 1, 5, and 15 minutes
 - The average system load is the average number of processes that are either runnable or uninterruptible.

- The average system load is the average number of processes that are either runnable (using or waiting to use the CPU) or uninterruptible (waiting for I/O operations, such as drive I/O). It is measured over the three time intervals. The average load is not scaled based on the number of CPUs in a system. On a single-CPU system, an average load of 1 indicates the system is always fully loaded, while on a quad-CPU system, the same value indicates the system is idle 75% of the time.

Viewing System Process Snapshots – ps

- The **ps** command displays all processes and sorts them by **%CPU** in descending order.

```
[root@openEuler ~]# ps -aux --sort -%cpu
USER      PID %CPU %MEM    VSZ RSS TTY      STAT START   TIME COMMAND
root          1  0.0  0.2  98988 10888 ?        Ss  11:57  0:00 /usr/lib/systemd/systemd --switched-root --system --deserialize 16
root          2  0.0  0.0      0   0 ?        S   11:57  0:00 [kthreadd]
root          3  0.0  0.0      0   0 ?        I<  11:57  0:00 [rcu_gp]
root          4  0.0  0.0      0   0 ?        I<  11:57  0:00 [rcu_par_gp]
root          6  0.0  0.0      0   0 ?        I<  11:57  0:00 [kworker/0:0H-events_highpri]
root          8  0.0  0.0      0   0 ?        I<  11:57  0:00 [mm_percpu_wq]
root          9  0.0  0.0      0   0 ?        S   11:57  0:00 [rcu_tasks_rude_]
```

- **USER:** process owner
- **PID:** process ID
- **%CPU:** percentage of CPU time used by a process
- **COMMAND:** command that started a process
- **TTY:** name of the terminal
- **START:** start time of a process
- **TIME:** total CPU time used by a process

top Command Output Overview

- Summary area:
 - System running duration and load
 - Process and CPU status
 - Memory usage
- Task area
 - Details of all processes organized by fields

| PID | PPID | TIME+ | %CPU | %MEM | PR | NI | S | VIRT | RES | UID | COMMAND |
|------|------|---------|------|------|----|----|---|--------|-------|-----|--------------------|
| 2515 | 2361 | 0:00.02 | 0.0 | 0.1 | 20 | 0 | R | 216520 | 3928 | 0 | top |
| 2361 | 2360 | 0:00.00 | 0.0 | 0.0 | 20 | 0 | S | 214160 | 3696 | 0 | bash |
| 2360 | 2256 | 0:00.00 | 0.0 | 0.1 | 20 | 0 | S | 15100 | 4888 | 0 | sshd |
| 2356 | 2 | 0:00.00 | 0.0 | 0.0 | 20 | 0 | I | 0 | 0 | 0 | kworker/3:3-events |
| 2353 | 2352 | 0:00.00 | 0.0 | 0.0 | 20 | 0 | S | 24188 | 3092 | 0 | (sd-pam) |
| 2352 | 1 | 0:00.01 | 0.0 | 0.1 | 20 | 0 | S | 20156 | 10092 | 0 | systemd |
| 2256 | 2190 | 0:00.01 | 0.0 | 0.1 | 20 | 0 | S | 15100 | 8748 | 0 | sshd |
| 2195 | 1 | 0:00.00 | 0.0 | 0.0 | 20 | 0 | S | 212540 | 2116 | 0 | agetty |
| 2194 | 1 | 0:00.00 | 0.0 | 0.0 | 20 | 0 | S | 212900 | 1684 | 0 | agetty |
| 2193 | 1 | 0:00.00 | 0.0 | 0.0 | 20 | 0 | S | 214632 | 2916 | 0 | crond |
| 2190 | 1 | 0:00.00 | 0.0 | 0.1 | 20 | 0 | S | 13448 | 6836 | 0 | sshd |

Viewing CPU Usage – top

```
%Cpu(s): 19.3 us, 4.0 sy, 0.0 ni, 74.7 id, 0.0 wa, 0.3 hi, 1.7 si, 0.0 st
```

- **%Cpu(s):** CPU status information, showing the usage percentage of each CPU
 - **us:** percentage of CPU time used by user-mode processes
 - **sy:** percentage of CPU time used by kernel-mode processes
 - **ni:** percentage of CPU time used by low-priority user-mode processes
 - **id:** percentage of idle CPU time
 - **wa:** percentage of the time that the CPU spends waiting for I/O operations
 - **hi:** percentage of CPU time used to handle hardware interrupts
 - **si:** percentage of CPU time used to handle software interrupts
 - **st:** percentage of time that a VM waits for the CPU

Viewing System Memory Usage – free

- The **free** command displays the system memory usage.

```
[root@openEuler ~]# free -m
total        used         free      shared  buff/cache   available
Mem:       3546          62        3338          16        145       3286
Swap:          0           0           0
```

- total:** total system memory
- used:** used memory
- free:** free memory
- shared:** memory shared by processes
- buffer/cache:** buffer/cache memory
- available:** memory available for starting new applications

Viewing Comprehensive System Information – vmstat

- The **vmstat** command displays information about the processes, memory, I/O, interrupts, and CPUs of the system.

```
[root@openEuler ~]# vmstat -a -S m
procs -----memory----- swap-- io-- system-- cpu-----
r b swpd free inact active si so bi bo in cs us sy id wa st
0 0 0 3467 110 74 0 0 32 1 14 16 0 0 100 0 0
```

- Memory metrics (**memory**)
 - swpd**: used swap memory
 - free**: free memory
 - buff**: memory used for buffers
 - cache**: memory used for the cache
 - inact**: inactive memory
 - active**: active memory
- Swap space metrics (**swap**)
 - si**: memory swapped from the drive per second
 - so**: memory swapped to the drive per second
- Process metrics (**procs**)
 - r**: number of runnable processes
 - b**: number of blocked processes waiting for I/Os

Viewing Comprehensive System Information – vmstat

- CPU metrics (**cpu**)
 - **us**: percentage of CPU time used by user processes
 - **sy**: percentage of CPU time used by system processes
 - **id**: percentage of idle CPU time
 - **wa**: percentage of the time that the CPU spends waiting for I/O operations
 - **st**: percentage of time that a VM is idle
 - **gu**: percentage of time that a VM is running
- I/O metrics (**io**)
 - **bi**: blocks read from block devices
 - **bo**: blocks written to block devices
- System interrupt metrics (**system**)
 - **in**: number of interrupts per second, including the clock
 - **cs**: number of context switches per second

Checking the System Cache – vmstat

- The **vmstat** command displays slabinfo of the system.

```
[root@openEuler ~]# vmstat -m
Cache           Num  Total   Size  Pages
nf_conntrack_expect      0     0    248    16
nf_conntrack          60     60   320    12
fat_inode_cache        42     42   768    21
fat_cache             204    204    40   102
fuse_request           0     0   152    26
fuse_inode             0     0   896    18
ext4_groupinfo_4k      308    308   144    28
```

- **Cache:** cache name
 - **Num:** number of active caches
 - **Total:** number of available caches
 - **Size:** cache size
 - **Pages:** number of pages

Checking Process Memory – top

- **Mib Mem** displays the physical memory information of the system.

| PID | USER | PR | NI | VIRT | RES | SHR | S | %CPU | %MEM | TIME+ | COMMAND |
|-----|------|----|-----|-------|-------|------|---|------|------|---------|-----------------------------|
| 1 | root | 20 | 0 | 99124 | 10852 | 8468 | S | 0.0 | 0.3 | 0:00.30 | 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-events_highpri |
| 8 | root | 0 | -20 | 0 | 0 | 0 | I | 0.0 | 0.0 | 0:00.00 | mm_percpu_wq |
| 9 | root | 20 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.00 | rcu_tasks_rude_ |
| 10 | root | 20 | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.00 | rcu_tasks_trace |

- **VIRT**: required virtual memory
- **RES**: actually used memory
- **SHR**: memory shared with other processes

VIRT



Checking Drive I/O Data – iostat

```
[root@openEuler ~]# iostat -dh
Linux 5.10.0-60.18.0.50.oe2203.x86_64 (openEuler)      05/18/2023      _x86_64_      (2 CPU)
          tps      kB_read/s      kB_wrtn/s      kB_dscd/s      kB_read      kB_wrtn      kB_dscd Device
        0.98       32.5k       1.2k        0.0k     146.4M       5.8M       0.0k  sda
```

- **Device:** device name
 - **tps:** I/O requests initiated to the device per second
 - **kB_read/s:** amount of data read from the device per second
 - **kB_wrtn/s:** amount of data written to the device per second
 - **kB_dscd/s:** amount of data discarded by the device per second
 - **kB_read:** total amount of read data
 - **kB_wrtn:** total amount of written data
 - **kB_dscd:** total amount of discarded data

Checking Block Device I/O Data – sar

- Syntax: sar -d [interval [count]]

- dev= *dev_list*: block device whose data is to be displayed

```
[root@openEuler ~]# sar -d 1 5
Linux 5.10.0-60.18.0.50.oe2203.x86_64 (openEuler)          05/18/2023      _x86_64_      (2 CPU)

07:11:13 AM    DEV    tps    rkB/s    wkB/s    dkB/s    areq-sz    aqu-sz    await    %util
07:11:14 AM    sda    2.00    0.00    24.00    0.00    12.00    0.00    1.00    0.20
07:11:15 AM    sda    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
07:11:16 AM    sda    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
07:11:17 AM    sda    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
07:11:18 AM    sda    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
Average:       sda    0.40    0.00    4.80    0.00    12.00    0.05    1.00    0.04
```

| Field | Description |
|-------|---|
| tps | Total number of transfers sent to the device per second |
| rkB/s | Number of kilobytes read from the device per second |
| wkB/s | Number of kilobytes written to the device per second |
| dkB/s | Number of kilobytes discarded by the device per second |

| Field | Description |
|---------|---|
| areq-sz | Average size of I/O requests sent to the device, in kilobytes |
| aqu-sz | Average queue length of requests sent to the device |
| await | Average time for sending I/O requests to devices that require services, in milliseconds |
| %util | Bandwidth usage of the device |

Checking the Network Connection Status – netstat

```
[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
tcp     0      276 192.168.0.48:22       119.3.119.20:3906   ESTABLISHED 1397/sshd: root [pr
                                         604/dhclient
udp    0      0 0.0.0.0:68              0.0.0.0:*
```

- **Proto:** protocol type
 - **Recv-Q:** received data volume (bytes)
 - **Send-Q:** sent data volume (bytes)
 - **Local Address:** local address and listening port
 - **Foreign Address:** remote address and listening port
 - **State:** port status
 - **PID/Program name:** process ID and name

Checking the Network Response Time – Ping

```
[root@openEuler ~]# ping -c 3 www.huaweicloud.com
PING koa8myv3.sslego-dk.tcloudscdn.com (123.139.99.5) 56(84) bytes of data.
64 bytes from 123.139.99.5 (123.139.99.5): icmp_seq=1 ttl=50 time=28.1 ms
64 bytes from 123.139.99.5 (123.139.99.5): icmp_seq=2 ttl=50 time=28.0 ms
64 bytes from 123.139.99.5 (123.139.99.5): icmp_seq=3 ttl=50 time=28.0 ms

--- koa8myv3.sslego-dk.tcloudscdn.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 27.971/28.031/28.111/0.058 ms
```

- **time**: response time
- **transmitted**: number of sent packets
- **receive**: number of received packets
- **loss**: number of lost packets
- **rtt**: round-trip delay
 - Minimum/average/maximum deviation

- **mdev** is short for Mean Deviation, indicating average deviation of the RTT of ICMP. A larger value means unstable network speed.

Checking the Network Bandwidth – iperf3

- The iperf3 tool is used to measure the bandwidth of the IP network and optimize parameters such as timing, protocol, and buffer.
- Command options

| Option | Description | Option | Description |
|---------------------|---|---------------|--|
| -s, --server | Enables the server mode. | -D, --daemon | Enables the server mode in the background. |
| -c, --client <host> | Enables the client mode and connects to the specified host. | -u, --udp | Uses UDP for the test. |
| -b, --bitrate | Limits the test bandwidth. | -t, --time | Specifies the test time. |
| -i, --interval | Specifies the interval for reporting throughput. | -R, --reverse | Conducts the reverse transmission test. |

- Syntax
 - iperf3 [-s|-c host] [options]
 - iperf3 [-h|--help] [-v|--version]

Checking Network Interface Traffic – sar

```
[root@openEuler ~]# sar -n DEV 2 1
Linux 5.10.0-60.18.0.50.oe2203.x86_64 (openEuler)          05/18/2023      _x86_64_      (2 CPU)

02:54:11 AM     IFACE    rxpck/s    txpck/s    rxkB/s    txb/s    rxcmp/s    txcmp/s    rxmcst/s    %ifutil
02:54:13 AM       lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
02:54:13 AM     ens3     1.00      0.50      0.06      0.08      0.00      0.00      0.00      0.00

Average:     IFACE    rxpck/s    txpck/s    rxkB/s    txb/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     1.00      0.50      0.06      0.08      0.00      0.00      0.00      0.00
```

- **IFACE:** network interface name
 - **rxpck/s:** total number of packets received per second
 - **txpck/s:** total number of packets transmitted per second
 - **rxkB/s:** total number of kilobytes received per second
 - **txkB/s:** total number of kilobytes transmitted per second
 - **rxcmp/s:** number of compressed packets received per second
 - **txcmp/s:** number of compressed packages transmitted per second
 - **rxmcst/s:** number of multicast packets received per second
 - **%ifutil:** network interface usage

Network Interface Failure Statistics – sar

```
[root@openEuler ~]# sar -n EDEV 2 1
Linux 5.10.0-60.18.0.50.oe2203.x86_64 (openEuler)      05/18/2023      _x86_64_      (2 CPU)

02:55:15 AM    IFACE    rxerr/s    txerr/s    coll/s    rxdrop/s    txdrop/s    txcarr/s    rxffram/s    rxfifo/s    txfifo/s
02:55:17 AM      lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
02:55:17 AM    ens3      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00

Average:    IFACE    rxerr/s    txerr/s    coll/s    rxdrop/s    txdrop/s    txcarr/s    rxffram/s    rxfifo/s    txfifo/s
Average:      lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
Average:    ens3      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
```

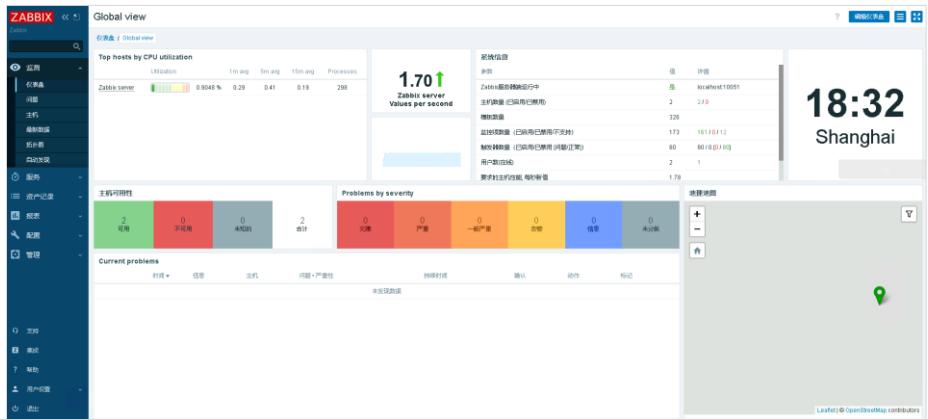
- **IFACE:** network interface name
 - **rxerr/s:** total number of bad packets received per second
 - **txerr/s:** total number of errors that occur per second during packet transmission
 - **coll/s:** number of conflicts per second during packet transmission
 - **rxdrop/s:** number of received packets discarded per second
 - **txdrop/s:** number of transmitted packets discarded per second
 - **txcarr/s:** number of carrier errors per second during packet transmission
 - **rxffram/s:** number of frame alignment errors per second during packet receiving
 - **rxfifo/s:** number of FIFO overrun errors per second during packet receiving
 - **txfifo/s:** number of FIFO overrun errors per second during packet transmission

Contents

1. Commands for Checking openEuler Performance
2. **System Monitoring with Zabbix**

Introduction to Zabbix

- Zabbix is an enterprise-class open-source monitoring solution that supports tens of thousands of servers, virtual machines, network devices, and millions of monitoring indicators.

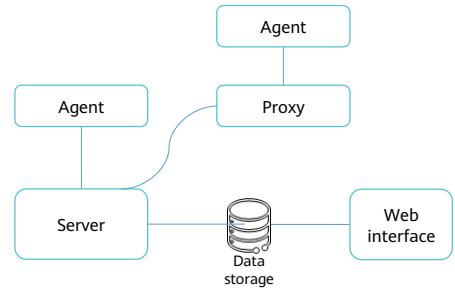


31 Huawei Confidential



Zabbix Architecture and Components

- Server
 - Zabbix server is a central component to which Zabbix agents report data on availability and integrity of systems, as well as statistics. It is also the central repository in which all configuration, statistical and operational data is stored.
- Data storage
 - All configuration and data collected by Zabbix are stored in the database.
- Web interface
 - Zabbix provides a Web-based interface for easy access from any platform. It is a part of the Zabbix server and usually (but not necessarily) runs on the same device as the Zabbix server.
- Proxy
 - Zabbix proxy can collect performance and availability data, essentially working on behalf of the server. Proxy is an optional part of Zabbix deployment and is used to implement the distributed architecture of Zabbix.
- Agent
 - Zabbix agent is deployed on a monitoring target to actively monitor local resources and applications, and reports collected data to Zabbix server for further processing.



- There are two agents in the current version: agent and agent2. Agent is developed using the C language and runs in an independent process. Agent2 is developed using the Go and C languages and runs in multi-thread mode on one process, reducing resource consumption and delivering high concurrency.

Zabbix Functions

- Zabbix is a highly integrated network monitoring solution that provides multiple functions in a software package.
 - Data collection
 - Flexible threshold definitions
 - Highly configurable alerting
 - Real-time graphing
 - Network monitoring
 - Extensive visualization options
 - Historical data storage
 - Easy configuration
 - Use of templates
 - Network discovery
 - Fast Web interface
 - Zabbix API
 - Permission system
 - Full featured and easily extensible agent
 - Binary daemons
 - Ready for complex environments

- For details, see
<https://www.zabbix.com/documentation/current/en/manual/introduction/features>

Working Principles of Zabbix

- Active mode
 - The agent actively communicates with the server and reports the current status of the monitoring target to the server.



- Passive mode
 - The server actively communicates with the agent and collects the current status of the monitoring target.



Zabbix Functional Areas

- **Monitoring:** displays data of monitoring targets. Any information collected, visualized, and operated by Zabbix is displayed in each component of the monitoring menu.
- **Services:** filters and monitors high-level services based on the current architecture.
- **Inventory:** provides an overview of host asset data based on selected parameters and allows users to view host asset details.
- **Reports:** contains various predefined and user-defined reports that mainly display system information, triggers, and collected data.
- **Configuration:** configures most Zabbix functions, such as hosts and host groups, data collection, data thresholds, issue notification, and data visualization.
- **Administration:** configure users, user groups, roles, and Zabbix management; open only to the super administrator.

Zabbix Basic Concepts

- Host and host group: A host is a networked device monitored by Zabbix through the network, and a host group is a logical grouping of hosts. A host group can contain hosts, templates, or both.
- Template: a set of entities (items, triggers, graphs, low-level discovery rules, web scenarios) ready to be applied to one or several hosts. It enables Zabbix to monitor hosts more quickly and easily.
- Trigger: a logical expression that defines a problem threshold and is used to "evaluate" data received in items. When received data are above the threshold, triggers go from **Ok** into a **Problem** state. When received data are below the threshold, triggers stay in/return to an **Ok** state.
- Event and action: An event is a single occurrence of something that deserves attention such as a trigger changing state or a discovery/agent autoregistration taking place. An action is a predefined means of reacting to an event and consists of conditions for triggering the event. The administrator can set event tags for event management, permission granulation, etc.

Zabbix Installation and Deployment

- Deploy LNMP or LAMP on the node where the web interface is located.
- Deploy a compatible database on the data storage node.
- Compile and install zabbix-server, and configure it according to your requirements.
- Install the web interface.
- Install zabbix-agent.

Zabbix Installation and Deployment – PHP Installation and Configuration

- Install PHP and its dependencies.

```
yum install -y php-cli php-fpm php-gd php-mbstring php-bcmath php-xml php-ldap php-mysqlnd
```

- Modify related 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
```

Zabbix Installation and Deployment – Zabbix Compilation Parameters

- During Zabbix compilation, add compilation parameters as required. Run the `./configure --help` command to view detailed information. Common parameters are as follows:
 - `--sysconfdir`: specifies the directory for storing system configurations. If this parameter is selected, you need to specify the path of the corresponding configuration file when starting `zabbix_server`.
 - `--enable-server/-enable-agent`: enables the server/agent.
 - `--with-mysql`: enables the support for the MySQL database. If another database is used, change `mysql` to the corresponding database name.
 - `--with-ssh2`: enables the SSH function.
 - `--with-net-snmp`: enables the SNMP function.
 - `--with-openipmi`: enables the IPMI function.
 - `--with-zlib, --with-libpthread, --with-libevent, --with-libpcre, --with-libcurl, --with-libxml2`: required dependencies.

Zabbix Installation and Deployment – Database Configuration

- Create a database for Zabbix and grant permissions to a specified user.

```
create database zabbix charset utf8 collate utf8_bin;
create user 'zabbix'@'%' identified by 'Huawei@123';
grant all on zabbix.* to 'zabbix'@'%';
flush privileges;
```

- Import required tables and data in sequence.

```
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
```

- Specify database information in the Zabbix configuration file.

```
DBHost=Database_IP_address
DBName=zabbix
DBUser=zabbix
DBPassword=Zabbix_password
DBPort=3306
```

Zabbix Installation and Deployment – Agent Installation

- Download the source file of the required version from the Yum repository.

```
rpm -Uvh https://repo.zabbix.com/zabbix/6.2/rhel/8/x86_64/zabbix-release-6.2-3.el8.noarch.rpm
```

- Install the agent.

```
yum install -y zabbix-agent2
```

- Specify the zabbix_server IP address.

```
sed -i 's/Server=127.0.0.1/Server=zabbix_server IP_address/g' /etc/zabbix/zabbix_agent2.conf
```

- After the agent is configured, start the agent.

Basic Zabbix Operations – Automatic Host Discovery

- Install and configure the agent on hosts to be monitored.
- Set auto discovery rules.
 - 1. Set the IP address range of hosts to be automatically discovered.
 - 2. Set the items to be checked.
 - 3. Set other configurations.
- Set the action after automatic discovery.
 - 4. In the **Action** area, bind conditions to automatic discovery rules.
 - 5. In **Operations** area, bind automatically discovered hosts to a host group and template.

This screenshot shows the 'Operations' configuration page. A red circle labeled '4' highlights the 'Name' field, which contains 'Auto add'. A red circle labeled '5' highlights the 'Conditions' section, specifically the row where 'Label' is 'A' and 'Name' is 'Discovery rule equals Discover'. Below this, there is a note: 'At least one operation must exist.' with 'Add' and 'Cancel' buttons.

This screenshot shows the 'Automatic Host Discovery' configuration page. A red circle labeled '1' highlights the 'IP range' field, which contains '192.168.0.1-254'. A red circle labeled '2' highlights the 'Checks' section, showing a table with one row and an 'Add' button. A red circle labeled '3' highlights the 'Device uniqueness criteria' section, where 'Host name' is selected. The 'Visible name' section also has 'Host name' selected. At the bottom, 'Enabled' is checked, and there are 'Add' and 'Cancel' buttons.

Basic Zabbix Operations – Manual Host Addition

- Install the agent on hosts to be monitored.
- Add a host on the web interface.
 - 1. Enter the host name, which must be the same as the host name on the OS.
 - 2. Link one or more templates to the host. A template contains monitoring items. If no template is selected, the added host will be unavailable.
 - 3. Select one or more host groups to which it belongs.
 - 4. Select one or more interface types for communication between the server and agent. The options include **Agent**, **SNMP**, **JMX**, and **IPMI**.

New host

Host IPMI Tags Macros Inventory Encryption Value mapping

* Host name (1)

Visible name

Templates (2) Select

* Host groups (3) Select

Interfaces (4) Add

Description

Monitored by proxy (4)

Enabled

- This course focuses on host monitoring for Linux, but Zabbix also supports Windows.
- The page for adding a host can be opened in the **Monitoring** area or **Configuration** area.
- When the interface is set to the agent, you need to enter the IP address (used for communicating with the server) and port (defaults to 10050) of the agent.

Basic Zabbix Operations – Monitoring Items Customization

- Agent configuration
 - (Optional) Write a script for obtaining the monitoring statistics.
 - Add related configuration files to the **zabbix_agent2** directory. Configuration file format:
 - **UserParameter=key.cmd**, where *key* indicates the key name, and *cmd* indicates the commands for obtaining the monitoring statistics or running the script for obtaining the monitoring statistics.
- Server configuration
 - Add monitoring items on the web interface.
 - 1. Specify the agent type.
 - 2. Specify the item key created on the agent.
 - 3. Select the host interface.
 - 4. Set the data retention period.
 - 5. Conduct a test.

- In the agent configuration file, add the commands for obtaining the monitoring statistics. Use a script if there are many commands.
- After configuring the agent, restart the agent for the configuration to take effect.

Basic Zabbix Operations – Trigger Customization

- You can customize triggers based on statistics returned by the monitoring items. The procedure is as follows:
 - 1. Set the required alarm severity of the trigger.
 - 2. Set conditions for triggering the alarm.
 - 3. Set conditions for clearing the alarm or restoring the system.

The screenshot shows the 'Trigger' configuration dialog in Zabbix. Key fields include:

- Name: []
- Event name: []
- Operational data: []
- Severity: Not classified (highlighted with a red circle)
- * Expression: (2) (highlighted with a red circle)
- Event generation: Expression (highlighted with a red circle)
- Generation mode: Single
- OK event closes: All problems (highlighted with a red circle)
- With manual close: []
- URL: []
- Description: []
- Enabled:



Basic Zabbix Operations – Graph Creation

- Zabbix can automatically generate graphs for monitoring items. It also allows customize graphs as follows:
 - 1. Specify the size of the graph to be created.
 - 2. Bind the graph with monitoring items.

The screenshot shows the 'Graph creation' dialog box in Zabbix. The 'Name' field is populated with 'Graph 1'. The 'Width' field is set to '900' and has a red circle around it. The 'Height' field is set to '200' and has a red circle around it. The 'Graph type' dropdown is set to 'Normal'. Under 'Show options', 'Show legend' and 'Show working time' are checked. Under 'Triggers', 'Show triggers' is checked. Under 'Percentiles', both 'Percentile line (left)' and 'Percentile line (right)' are unchecked. Under 'Y-axis', both 'Y axis MIN value' and 'Y axis MAX value' are set to 'Calculated'. The 'Items' section contains a table with one row. The first column is 'Items' with a red circle around the 'Add' button. The second column is 'Name' with an empty input field. The third column is 'Function' with an empty input field. The fourth column is 'Draw type' with an empty input field. At the bottom are 'Add' and 'Cancel' buttons, with 'Add' having a red circle around it.

Quiz

1. (True or False) Zabbix allows automatically discovering or manually adding hosts.
 - A. True
 - B. False
2. (Multiple-answer question) Which of the following commands can be used to view system memory usage?
 - A. top
 - B. ps
 - C. sar
 - D. mpstat

- A
- ABC

Summary

- This course describes basic commands for checking system performance, mainstream system monitoring tool Zabbix.

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

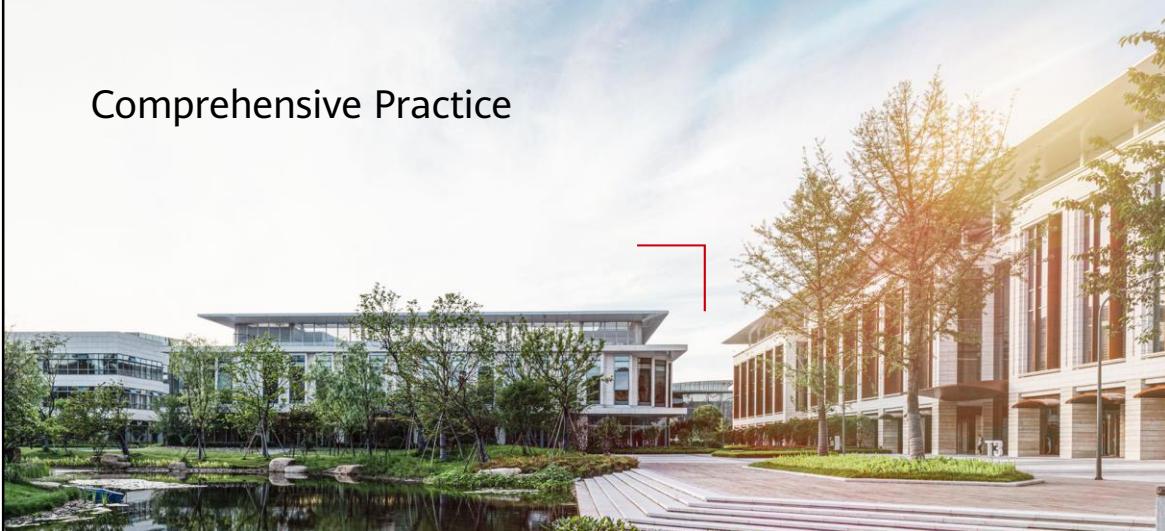
Bring digital to every person, home, and
organization for a fully connected,
intelligent world.

Copyright©2024 Huawei Technologies Co., Ltd.
All Rights Reserved.

The information in this document may contain predictive
statements including, without limitation, statements regarding
the future financial and operating results, future product
portfolio, new technology, etc. There are a number of factors that
could cause actual results and developments to differ materially
from those expressed or implied in the predictive statements.
Therefore, such information is provided for reference purpose
only and constitutes neither an offer nor an acceptance. Huawei
may change the information at any time without notice.



Comprehensive Practice



Foreword

- In previous courses, we covered LAMP/LNMP architecture, cluster architecture, automatic O&M, and system monitoring. This course presents practical applications of this knowledge within simulated work environments.

Objectives

- Upon completion of this course, you will understand:
 - LANP/LNMP architecture in actual working scenarios.
 - Application deployment in actual working scenarios.

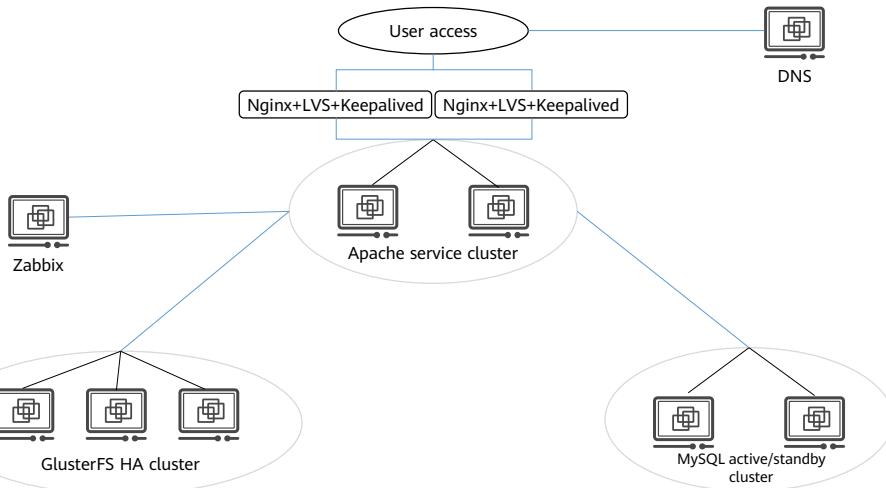
Contents

- 1. Project Requirements**
2. Project Instructions

Project Requirements

- Assume that an enterprise plans to use WordPress to build a blog system. To ensure the system's stability, the administrator runs the system on a web service cluster consisting of two Apache servers.
- In addition to load WordPress, web servers need to display commonly used images for employees on web pages.
- The back-end database is MySQL which needs to be deployed in active/standby mode to ensure high data availability (HA).
- Data and static images of the blog system are stored in the shared file system provided by GlusterFS.
- Considering the system HA, a proxy cluster needs to be configured, which should consist of Nginx, LVS, and Keepalived at the front end of the system cluster.
- To facilitate usage, services need to be accessed using domain names that are resolved by the intranet DNS.
- Hosts involved in the new system need to be monitored by the service monitoring system.
- Ansible Playbook is used for automatic deployment.

Service Architecture Design



Service IP Address Planning

| Service | Host | IP Address | Description |
|---|------------|--------------|-------------------------|
| Nginx+LVS+Keepalive cluster Floating IP address: 192.168.1.10 | Nginx-01 | 192.168.1.11 | Service plane |
| | | 10.0.0.11 | Data and storage planes |
| | Nginx-02 | 192.168.1.12 | Service plane |
| | | 10.0.0.12 | Data and storage planes |
| GlusterFS cluster Floating IP address: 10.0.0.20 | Gluster-01 | 10.0.0.21 | Storage plane |
| | Gluster-02 | 10.0.0.22 | Storage plane |
| | Gluster-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 |
| Zabbix | Zabbix | 192.168.1.15 | Service plane |
| | | 10.0.0.15 | Management plane |
| Ansible | Ansible | 192.168.1.50 | Management plane |

Other Service Planning

| Service | Database Name | User | Password |
|-----------|---------------|------|------------|
| WordPress | WP | wp | Huawei@123 |

| Service | Volume | Capacity | Remarks |
|--------------|--------|----------|---------|
| WordPress | wp | 10G | |
| Display page | image | 10G | |

| Service | Domain Name | Remarks |
|--------------|---------------------|---------|
| WordPress | www.test.com/blog/ | |
| Display page | www.test.com/image/ | |

Discussion: Procedure

- () Install basic software, including MySQL, Apache, Nginx, GlusterFS, LVS, Keepalived, and PHP.
- () Configure the database. Specifically, build a MySQL active/standby cluster and create the database required by WordPress.
- () Configure GlusterFS. Specifically, create two logical volumes, one for WordPress and the other for static images.
- () Configure Apache. Mount the logical volume created to Apache and create a virtual host. Access WordPress through port 81 and access static images through port 82.
- () Configure LVS and Keepalived to provide a layer-4 proxy for Nginx.
- () Configure Nginx to provide a layer-7 proxy for Apache. When a user accesses "192.168.1.10/blog", the user is redirected to WordPress. When a user accesses "192.168.1.10/image", the user is redirected to the page displaying static images.
- () Configure the DNS to resolve www.test.com to 192.168.1.10.
- () Add all involved hosts to Zabbix.

- Sort the provided steps. If the listed steps cannot cover all operations, add other required steps.

Procedure

1. Install basic software, including MySQL, Apache, Nginx, GlusterFS, LVS, Keepalived, and PHP.
2. Configure the database. Specifically, build a MySQL active/standby cluster and create the database required by WordPress.
3. Configure GlusterFS. Specifically, create two logical volumes, one for WordPress and the other for static images.
4. Configure Apache. Mount the logical volume created to Apache and create a virtual host. Access WordPress through port 81 and access static images through port 82.
5. Configure LVS and Keepalived to provide a layer-4 proxy for Nginx.
6. Configure Nginx to provide a layer-7 proxy for Apache. When a user accesses "192.168.1.10/blog", the user is redirected to WordPress. When a user accesses "192.168.1.10/image", the user is redirected to the page displaying static images.
7. Configure the DNS to resolve www.test.com to 192.168.1.10.
8. Add all involved hosts to Zabbix.

- The answer here is for reference only. You can sort these steps in any sequence as long as your procedure is feasible.

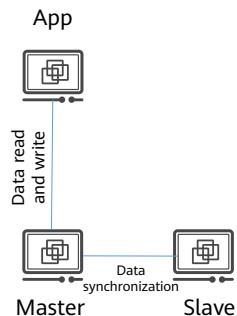
Contents

1. Project Requirements
2. **Project Instructions**

Other Operations – MySQL Active/Standy Cluster Setup

- Master node configurations

- Create a user for the slave node to synchronize data:
 - create user slave identified with mysql_native_password by 'Huawei@123';
 - Assign permissions to the created user:
 - GRANT REPLICATION SLAVE ON *.* to 'slave'@'%';
 - FLUSH PRIVILEGES;
 - Modify the **my.cnf** configuration file and add **server-id** and **log-bin** configurations:
 - server-id=1
 - log-bin=/var/lib/mysql/binlog
 - Check the binary log name and offset:
 - show master status;
- Slave node configurations
- Modify the **my.cnf** configuration file and add **server-id** and **log-bin** configurations:
 - server-id=2
 - log-bin=/var/lib/mysql/binlog
 - Specify information about the master node.



- Command that specifies information about the master node:
 - **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;

Quiz

1. (Discussion) What can be improved in the practice described in this chapter?

- Reference:
 - 1. Security-related policies are not configured.
 - 2. There are many Ansible playbooks, which can be configured in a unified manner using roles.
 - 3. LVS load balancing is not implemented in the Keepalived+LVS+Nginx cluster. You are advised to remove Nginx from the cluster.

Summary

- Based on the preceding courses, this course provides a practice in a simulated working scenario to help you understand the key steps about implementing the LAMP/LNMP architecture.

Acronyms

| Acronym | Full Spelling | Description |
|---------|-----------------------|------------------|
| LNMP | Linux\Nginx\MySQL\PHP | A software stack |
| LVS | Linux Virtual Server | |

Thank you.

把数字世界带入每个人、每个家庭、

每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and organization for a fully connected, intelligent world.

Copyright©2024 Huawei Technologies Co., Ltd.
All Rights Reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.

