Installation

Installation JumpServer Enterprise Edition

To begin, you need to request the latest distribution file from us via email at info@afi-d.com or on Telegram: @mapceaheh.

1.System Requirements:

- OS: Linux/AMD64 (arm64) x86_64(aarch64) kernel version 4.0 or higher (preferably Redhat, Debian, Ubuntu families)
- CPU: 4 cores
- RAM: 8 GB
- HDD: 60 GB

2. Installation

Place the downloaded file in the directory **/opt**

Execute the following commands (file names may vary with new versions):

```
$ cd /opt
$ tar -xf jumpserver-offline-installer-v3.10.3-amd64.tar.gz
$ cd jumpserver-offline-installer-v3.10.3-amd64
```

Next, you can edit the configuration file to change installation parameters, for example, to use an external MySQL database or change the installation folder.

```
# nano /opt/jumpserver-offline-installer-v3.10.3-amd64/config-example.txt
```

Start the installation:

```
# cd jumpserver-offline-installer-v3.10.3-amd64
# ./jmsctl.sh install
```

During installation, you will need to confirm the data entered in the configuration file or provide other data if you did not fill in the configuration file in advance.

Start the application:

```
# ./jmsctl.sh start
```

3. Start application

Navigate to the product folder (the folder name may change with new versions) and start application:

```
# cd /opt/jumpserver-installer-v3.10.3
# ./jmsctl.sh start
```

After this, you can access the web interface at:

http://IP/ Login: **admin**
Password: **ChangeMe**

and begin configuring the system.

Installation JumpServer Community Edition

Attention: *You will not be able to install a license to activate Enterprise (x-pack) features in the Community Edition. If you plan to do PoC of Enterprise version, follow the installation instructions for JumpServer Enterprise Edition.*

1. Server preparation

System Requirements:

- OS: Linux/AMD64 (arm64) x86_64(aarch64) kernel version 4.0 or higher
- CPU: 4 cores
- RAM: 8 GB
- HDD: 60 GB

Installation of additional components on Debian\Ubuntu as an example:

```
# apt-get update
# apt-get install -y wget curl tar gettext iptables
```

2. JumpServer Installation

Quick Online JumpServer Installation:

In this case, JumpServer will be installed with default parameters, and MySQL and Redis databases will be installed in containers on the same server.

```
$ curl -sSL https://github.com/jumpserver/jumpserver/releases/latest/download/quick_start.sh | bash
```

Wait for the script execution to complete.

Standard Online Installation:

Download the latest installer from GitHub: https://github.com/jumpserver/installer/releases/ Below are example commands for version 3.10.3:

```
# cd /opt/
# wget https://github.com/jumpserver/installer/releases/download/v3.10.3/jumpserver-installer-v3.10.3.t
# tar -xf jumpserver-installer-v3.10.3.tar.gz
```

Next, you can edit the configuration file to change installation parameters, for example, to use an external MySQL database or change the installation folder for cluster installation etc.

```
# nano /opt/jumpserver-installer-v3.10.3/config-example.txt
```

Start the installation:

```
# cd ./jumpserver-installer-v3.10.3
# ./jmsctl.sh install
```

During installation, you will need to confirm the data entered in the configuration file or provide other data if you did not fill in the configuration file in advance.

3. Starting the Application

After the installation is complete, navigate to the product folder (the folder name may change with new versions) and start the application:

```
$ cd /opt/jumpserver-installer-v3.10.3
# ./jmsctl.sh start
```

After that, you can access the web interface at:

http://IP/ Login: **admin**
Password: **ChangeMe**

and begin configuring the system.

Operation and Maintenance with command line jmsctl

Operation and Maintenance - jmsctl

JumpServer includes a built-in command-line tool for operation and maintenance by default - **jmsctl.** To view the help documentation, run the command:

```
jmsctl help
```

JumpServer Application Management:

```
./jmsctl.sh [COMMAND] [ARGS...]
./jmsctl.sh --help
```

Installation Commands:

- install - Install the JumpServer service

Management Commands:

- config - Configure the tool, run jmsctl config --help to view help
- start - Start the JumpServer service
- stop - Stop the JumpServer service
- restart - Restart the JumpServer service
- status - Check the status of the JumpServer service
- down - Stop the JumpServer service
- uninstall - Uninstall the JumpServer service

Additional Commands:

- load_image - Load a Docker image
- backup_db - Backup the JumpServer database
- restore_db [file] - Restore data from a database backup file
- raw - Execute a docker compose command
- tail [service] - View service logs

JumpServer port discription

List of Network Ports

JumpServer requires the following network ports to be open for proper operation. Administrators can open the appropriate ports in the network and on the host depending on the deployment scheme of JumpServer components.

Port	Purpose	Description
22	SSH	Installation, updates, and management
80	Web HTTP Service	Access to the JumpServer web interface via HTTP
443	Web HTTPS Service	Access to the JumpServer web interface via HTTPS
3306	Database Service	Used by MySQL
6379	Database Service	Used by Redis
3389	Razor Service Port	Connection to Windows assets via RDP Client
2222	SSH Client	Connection to JumpServer via terminal tools (Xshell, PuTTY, etc.)
33061	Magnus MySQL Service Port	Connection to MySQL via DB Client
33062	Magnus MariaDB Service Port	Connection to MariaDB via DB Client
54320	Magnus PostgreSQL Port	Connection to PostgreSQL via DB Client
63790	Magnus Redis Port	Connection to Redis via DB Client
30000-30100	Magnus Oracle Ports	Connection to Oracle via DB Client, port range can be configured

Installing SSL Certificates and Configuring HTTPS

What is the Purpose of JumpServer Reverse Proxy?

Nginx supports secure WebSockets (wss://), managing connections and securing the channel with an SSL certificate. To enable the copy-paste functionality in the RDP protocol, a trusted SSL certificate must be deployed. Copy-paste in RDP assets is only possible when accessed via the HTTPS protocol.

Installing SSL Certificates and Configuring HTTPS for the Web Interface

Prepare an SSL certificate (note that the certificate **must be in PEM format**). Certificates should be placed in the directory **/opt/jumpserver/config/nginx/cert**

Stop the JumpServer service:

```
./jmsctl.sh stop
```

Open the JumpServer configuration file:

```
vi /opt/jumpserver/config/config.txt
```

Find and update the Nginx configuration parameters:

```
## Nginx Configuration
HTTP_PORT=80
SSH_PORT=2222
RDP_PORT=3389

## HTTPS Configuration
HTTPS_PORT=443              # External port for HTTPS, default is 443
SERVER_NAME=www.domain.com   # Your domain for HTTPS
SSL_CERTIFICATE=xxx.pem      # Your certificate name in /opt/jumpserver/config/nginx/cert
SSL_CERTIFICATE_KEY=xxx.key  # Your key file name in /opt/jumpserver/config/nginx/cert
```

Save the configuration changes and start JumpServer:

```
./jmsctl.sh start
```

If you need to further edit the Nginx configuration file:

```
vi /opt/jumpserver/config/nginx/lb_http_server.conf
```

Multi-Level Reverse Proxy on Nginx

Hint:

This configuration is suitable when there is a shared external proxy server at the top level. It is an example of multi-level reverse proxying on Nginx. Each proxy section must be configured to support long WebSocket connections.

Editing the Configuration File:

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

Example Configuration without SSL:

```
server {

    listen 80;
    server_name demo.jumpserver.org;  # Replace with your domain

    client_max_body_size 4096m;  # Limit for maximum file upload size

    location / {
        # Specify the IP address of the JumpServer Nginx server
        proxy_pass http://192.168.244.144;
        proxy_http_version 1.1;
        proxy_buffering off;
        proxy_request_buffering off;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $remote_addr;
    }
}
```

Recommendation:

For more secure access, it is recommended to configure SSL and use the HTTPS protocol, following the guidelines from Mozilla SSL Configuration Generator.

Example Configuration with SSL:

Redirecting HTTP to HTTPS:

```
server {
    listen 80;
    server_name demo.jumpserver.org;  # Replace with your domain
    return 301 https://$server_name$request_uri;  # Redirect all HTTP requests to HTTPS
}
```

Configuring HTTPS:

```
server {
    listen 443 ssl http2;
    server_name demo.jumpserver.org;  # Replace with your domain
    ssl_certificate sslkey/1_jumpserver.org_bundle.crt;  # Path to your SSL certificate
    ssl_certificate_key sslkey/2_jumpserver.org_bundle.key;  # Path to your certificate key
    ssl_session_timeout 1d;
    ssl_session_cache shared:MozSSL:10m;
    ssl_ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-
RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384;
    ssl_prefer_server_ciphers off;
    ssl_protocols TLSv1.1 TLSv1.2;
    add_header Strict-Transport-Security "max-age=63072000" always;

    client_max_body_size 4096m;  # Limit for maximum file upload size

    location / {
        # Specify the IP address of the JumpServer Nginx server
        proxy_pass http://192.168.244.144;
        proxy_http_version 1.1;
        proxy_buffering off;
        proxy_request_buffering off;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $remote_addr;
    }
}
```

3. Other Load Balancers (SLB)

Hint:

1. Correctly configure long WebSocket connection support.
2. Consider session management issues.

HAProxy configuration for JumpServer HA-cluster

HAProxy (High Availability Proxy) — is an **open-source software tool** used for load balancing and traffic proxying at the network protocol level, typically employed to distribute traffic across multiple servers. It is one of the most popular solutions for enhancing the availability and performance of web applications and services.

To install HAProxy on Ubuntu:

```
sudo apt install haproxy -y
```

After installation, you need to edit the configuration file, which is the main aspect of setting up HAProxy. The configuration file is typically located at **/etc/haproxy/haproxy.cfg**.

Example Configuration File from the Vendor's Documentation:

```
global
    log         127.0.0.1 local2
    chroot      /var/lib/haproxy
    pidfile     /var/run/haproxy.pid
    maxconn     4000
    user        haproxy
    group       haproxy
    daemon
    stats socket /var/lib/haproxy/stats

defaults
    log                     global
    option                  dontlognull
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000

listen stats
    bind *:8080
    mode http
```

```
    stats enable
    stats uri /haproxy
    stats refresh 5s
    stats realm haproxy-status
    stats auth admin:password

listen jms-web
    bind *:80
    mode http
    option httpchk GET /api/health/
    stick-table type ip size 200k expire 30m
    stick on src
    balance leastconn
    server 192.168.100.21 192.168.100.21:80 weight 1 cookie web01 check inter 2s rise 2 fall 3
    server 192.168.100.22 192.168.100.22:80 weight 1 cookie web02 check inter 2s rise 2 fall 3
```

After modifying the configuration file, restart and enable HAProxy:

```
systemctl enable haproxy
systemctl start haproxy
```

JumpServer HA-cluster configuration

JumpServer (including Community Edition) fully supports HA clustering without any restrictions. In this article, I will show how it works.

Why is an HA cluster needed for JumpServer

An HA (High Availability) cluster for JumpServer is necessary to ensure high system availability and minimize downtime. It enables:

1. **Avoiding downtime**: If one cluster node fails, another continues to handle user requests.
2. **Ensuring fault tolerance**: Clustering allows the system to automatically switch to available nodes in case of errors.
3. **Improving performance**: Load is distributed across multiple nodes, enhancing system responsiveness under high user demand.
4. **Increasing data storage reliability**: Using shared resources like MySQL and Redis with clustering support minimizes the risk of data loss.
5. **Scalability**: The cluster can be easily expanded by adding more nodes to handle more users and tasks.

This architecture is particularly important for organizations where JumpServer is used as a critical access and control system.

Components of the JumpServer Cluster

Nodes/JumpServer Nodes - Core cluster nodes with JumpServer installed. Each server does not store "useful" data, making it possible to clone, copy, delete, or add nodes as needed.

Database MySQL/PostgreSQL - The main DBMS for storing all JumpServer data, including system settings, device parameters, user accounts, and passwords for target systems. By default, it also stores session text logs such as SSH commands, SQL queries, and keyboard input in RDP sessions.
By default, JumpServer creates and uses PostgreSQL within a container on the same server where JumpServer is installed.

Redis Database - An auxiliary database for caching. It can be a shared database for the entire cluster or separate databases for each cluster node.
By default, JumpServer creates and uses Redis within a container on the same server where JumpServer is installed.

Video Recordings Storage - By default, it stores session recordings in the folder **$folder/core/data/media**, where **$folder** is specified in the main configuration file (default: **VOLUME_DIR=/data/jumpserver**). The product's web interface allows setting up external video storage: **SFTP, S3, Ceph, MinIo, and others.**

Command Logs Storage - Logs are stored in the main database by default. The web interface allows configuring log storage in **Elasticsearch**.

Load Balancer - Usually based on HAProxy, but other options can be used.

Cluster Architecture

Typically, a JumpServer cluster consists of two or more cluster nodes that:

- Are connected to a shared MySQL/PostgreSQL database (or cluster).
- Are connected to a shared Redis database (or each has its own).
- Share a common storage for video recordings:
  - A common folder **$folder/core/data/** (usually implemented via an NFS server).
  OR
  - External storage for session recordings (e.g., SFTP, S3, Ceph, MinIo).
- Share a common storage for command logs:
  - Based on a shared database (default).
  OR
  - Based on Elasticsearch.
- Use a load balancer (optional).

Example of creating a JumpServer HA Cluster with two nodes

Example of creating a JumpServer cluster
- With a shared MySQL database
- With a shared Redis database
- With a shared folder for storing video recordings **$folder/core/data/** using an NFS server.

For this, we need:

1. **Server with NFS, MySQL, Redis**:
   - 4 CPUs, 8 GB of RAM.
2. **JumpServer Node1**:
   - 4 CPUs, 8 GB of RAM, 100 GB of free disk space.
3. **JumpServer Node2**:
   - 4 CPUs, 8 GB of RAM, 100 GB of free disk space.
4. **HAProxy Server** (or another load balancer).

---

1. Preparing the server with NFS, MySQL, and Redis

- **Server**: Ubuntu 22.04, IP: `10.10.50.10`

Installing and configuring NFS

Commands may vary for different Linux versions, but generally, you need to create a shared folder:

```
sudo apt install nfs-kernel-server
sudo mkdir -p /data
```

```
sudo chown -R nobody:nogroup /data/
sudo chmod 777 /data/
sudo nano /etc/exports
```

Add the following line to the `/etc/exports` file:

```
/data 10.10.50.10/24(rw,sync,no_subtree_check)
```

Apply the settings and restart the NFS service:

```
sudo exportfs -a
sudo systemctl restart nfs-kernel-server
```

Installing and configuring MySQL

Instructions depend on the OS version. To create a database and user, run the following commands:

```
mysql -uroot
mysql> create database jumpserver default charset 'utf8';
mysql> set global validate_password_policy=LOW;
mysql> create user 'jumpserver'@'%' identified by 'KXOeyNgDeTdpeu9q';
mysql> grant all on jumpserver.* to 'jumpserver'@'%';
mysql> flush privileges;
mysql> exit;
```

Don’t forget to configure the firewall to open the MySQL port (`3306`).

Installing and configuring Redis

Instructions depend on the OS version. After installing Redis, run the following commands:

```
sed -i "s/bind 127.0.0.1/bind 0.0.0.0/g" /etc/redis.conf
sed -i "561i maxmemory-policy allkeys-lru" /etc/redis.conf
sed -i "481i requirepass KXOeyNgDeTdpeu9q" /etc/redis.conf
```

This will allow access to Redis with the password `KXOeyNgDeTdpeu9q`. Make sure to use a unique password for your server. Open the port `6379` in the firewall.

2. Installing JumpServer

Installing the first JumpServer node

Mounting the NFS Directory

Install the NFS client, mount the folder, and configure automatic mounting at startup:

```
sudo apt install nfs-common
mkdir -p /opt/jumpserver/core/data
```

```
mount -t nfs 10.10.50.10:/data /opt/jumpserver/core/data
echo "10.10.50.10:/data /opt/jumpserver/core/data nfs defaults 0 0" >> /etc/fstab
```

Configuring JumpServer

Edit the `config-example.txt` file in the installer directory:

```
# Modify the following parameters, leave others as default.
# IMPORTANT: SECRET_KEY must match on all JumpServer nodes, or the data will not decrypt.

VOLUME_DIR=/opt/jumpserver

SECRET_KEY=
BOOTSTRAP_TOKEN=
LOG_LEVEL=ERROR
SESSION_EXPIRE_AT_BROWSER_CLOSE=True

# MySQL

DB_HOST=10.10.50.10
DB_PORT=3306
DB_USER=jumpserver
DB_PASSWORD=KXOeyNgDeTdpeu9q
DB_NAME=jumpserver

# Redis

REDIS_HOST=10.10.50.10
REDIS_PORT=6379
REDIS_PASSWORD=KXOeyNgDeTdpeu9q

# KoKo Lion
SHARE_ROOM_TYPE=redis
REUSE_CONNECTION=False
```

Run the installation:

```
./jmsctl.sh install
```

After the installation is complete, you will receive the following values:

```
SECRET_KEY=kWQdmdCQKjaWlHYpPhkNQDkfaRulM6YnHctsHLlSPs8287o2kW
BOOTSTRAP_TOKEN=KXOeyNgDeTdpeu9q
```

Installing the Second JumpServer Node

Install the NFS client and mount the folder just as on the first node.

When editing the JumpServer configuration file, **fill in the values for `SECRET_KEY` and `BOOTSTRAP_TOKEN`** obtained after installing the first node:

```
VOLUME_DIR=/opt/jumpserver

SECRET_KEY=kWQdmdCQKjaWlHYpPhkNQDkfaRulM6YnHctsHLlSPs8287o2kW
BOOTSTRAP_TOKEN=KXOeyNgDeTdpeu9q
LOG_LEVEL=ERROR
SESSION_EXPIRE_AT_BROWSER_CLOSE=True

# MySQL

DB_HOST=10.10.50.10
DB_PORT=3306
DB_USER=jumpserver
DB_PASSWORD=KXOeyNgDeTdpeu9q
DB_NAME=jumpserver

# Redis

REDIS_HOST=10.10.50.10
REDIS_PORT=6379
REDIS_PASSWORD=KXOeyNgDeTdpeu9q

# KoKo Lion
SHARE_ROOM_TYPE=redis
REUSE_CONNECTION=False
```

And run the installation:

```
./jmsctl.sh install
```

Result

After completing the setup, you will have two JumpServer nodes sharing one MySQL/Redis server and NFS storage. You can use any of the nodes to access target devices or configure HAProxy to automatically redirect users to an active node.