




Linux





- [Mosh](#)
- [Supervisor](#) [Rocky Linux](#)
- [Rsyncd](#) [Rocky Linux](#)
- [Rocky Linux](#) [Docker](#) [YAPI](#)
- [SSH](#)
- [cURL](#)
- [Docker](#) [ShowDoc](#)

Mosh

Mosh  Mobile Shell  SSH


Mosh

Mosh 

- 
- 
- 
- 

Mosh

1. Linux

```
# Ubuntu/Debian
sudo apt-get install mosh

# CentOS/RHEL
sudo yum install mosh

# Fedora
sudo dnf install mosh
```

2. macOS

```
brew install mosh
```

3. Windows

Windows 安装

- 1. 安装 WSL 和 Cygwin
- 2. 安装 mosh
- 3. 安装 mosh 和 Windows Terminal



1. 安装

```
mosh username@remote_host
```

2. 安装

```
mosh --ssh="ssh -p 2222" username@remote_host
```

3. 安装 SSH 代理

```
mosh --ssh="ssh -i ~/.ssh/id_rsa" username@remote_host
```



选项	说明
-p PORT	指定 UDP 端口范围 60000:61000
--predict=never	从不预测
--predict=always	总是预测
--predict=adaptive	自适应预测
--ssh="COMMAND"	指定 SSH 命令
-a	指定 IP 地址
-n	指定 SSH 代理



1.

```
mosh --ssh="ssh -J jump_user@jump_host" user@target_host
```

2.

```
#  tmux  screen  mosh  
mosh user@host -- tmux new -A -s session_name
```

3.

mosh-server

```
mosh --server="/usr/local/bin/mosh-server" user@host
```

Mosh SSH

- SSH TCP
 - Mosh UDP 60000-61000
- SSH
 - Mosh
- SSH
 - Mosh






Mosh UDP

```
#  UDP 60000-61000   
sudo ufw allow 60000:61000/udp
```



1.


```
#  mosh
ssh user@host "which mosh-server"

#  UDP 
telnet host 60000
```

2.

```
#  UTF-8 
mosh --ssh="ssh -o SendEnv=LC_*" user@host
```




3.

```
# 
mosh --ssh="ssh -C" user@host
```



Mosh

 SSH 

1.   
2.  

Mosh



```
mosh --help
```

Mosh ☐☐☐ SSH

[illegible]

--	--	--	--	--	--	--

Supervisor 在 Rocky Linux



Supervisor 在 Python 在 UNIX
在 Rocky Linux 在 Supervisor 在

在 Supervisor

1. 在 EPEL 在

```
sudo dnf install epel-release  
sudo dnf update
```

2. 在 Supervisor

```
sudo dnf install supervisor
```

3. 在

```
sudo systemctl enable supervisord  
sudo systemctl start supervisord
```

4. 在

```
sudo supervisorctl status
```



1. [][][][][][][][]

[][][][][][] [etc/supervisord.conf]

2. [][][][][][][][]

[][][][][][] [etc/supervisord.d/][][][][][][] .ini [][][]

```
[include]
files = /etc/supervisord.d/*.ini
```

[][][][][][][][][][]

1. [][][][][][][][][][]

[[etc/supervisord.d/][][][][][][][][][][] myapp.ini]

```
[program:myapp]
command=/usr/bin/python3 /path/to/your/app.py
directory=/path/to/your/app
user=yourusername
autostart=true
autorestart=true
startretries=3
stderr_logfile=/var/log/myapp.err.log
stdout_logfile=/var/log/myapp.out.log
environment=PYTHONPATH="/path/to/your/app"
```

2. [][][][][][][]

[][]	[][]
command	[][][][][][]
directory	[][][][][][][][][][]
user	[][][][][][]

配置項目	設定
autostart	Supervisor 起動
autorestart	(true/false/unexpected)
startretries	
stderr_logfile	
stdout_logfile	
environment	

Supervisor 操作

1. 再起動

```
sudo supervisorctl reread
sudo supervisorctl update
```

2. 起動

```
# 起動
sudo supervisorctl start myapp

# 停止
sudo supervisorctl stop myapp

# 再起動
sudo supervisorctl restart myapp

# ステータス確認
sudo supervisorctl status

# 特定のプログラムのステータス確認
sudo supervisorctl status myapp
```

3. 終了

```
# 查看
```

```
sudo tail -f /var/log/supervisor/supervisord.log
```

```
# 查看
```

```
sudo tail -f /var/log/myapp.out.log
```

Web 应用

1. Web 应用

```
vim /etc/supervisord.conf
```

```
[inet_http_server]
port=*:9001
username=admin
password=yourpassword
```

2. Supervisor

```
sudo systemctl restart supervisord
```

3. Web 应用

```
curl http://your-server-ip:9001
```

```
curl http://your-server-ip:9001
```

1. 配置

```
[group:mygroup]
programs=myapp1,myapp2
priority=999
```

2. 配置

```
[eventlistener:memmon]
command=memmon -p myapp=200MB
events=TICK_60
```

3. 配置

在 `/etc/logrotate.d/supervisor` 中

```
/var/log/supervisor/*.log {
    daily
    missingok
    rotate 14
    compress
    delaycompress
    notifempty
    create 0640 root root
    postrotate
        /usr/bin/supervisorctl pid > /dev/null && /usr/bin/supervisorctl reopen_logs
    endscript
}
```

配置

1. 配置

- 配置 `/etc/logrotate.d/supervisor`

```
sudo supervisorctl reread
sudo supervisorctl update
```
- 配置 `/var/log/supervisor/supervisord.log` 文件
- 配置 `/etc/logrotate.d/supervisor` 文件

2. 配置

- ☐
- ☐

1. ☐ root ☐
2. Web ☐
3. ☐ Web ☐ IP
4. ☐
5. ☐ Supervisor ☐

☐ Rocky Linux ☐ Supervisor☐

Rsyncd on Rocky Linux



Rsyncd on Rsync on Rocky Linux 8/9 Rsyncd

Getting Rsyncd

1. Install Rsync

```
sudo dnf install rsync
```

2. Verify Installation

```
rsync --version
```

Configuring Rsyncd

1. Edit Configuration File

```
sudo vi /etc/rsyncd.conf
```

2. Set Permissions

```
# Set permissions
uid = nobody
gid = nobody
```

```
use chroot = yes
max connections = 4
pid file = /var/run/rsyncd.pid
lock file = /var/run/rsync.lock
log file = /var/log/rsyncd.log
timeout = 300

# [ ] [ ] [ ] [ ]
[backup]
    path = /data/backup
    comment = Backup Directory
    read only = no
    list = yes
    auth users = rsyncuser
    secrets file = /etc/rsyncd.secrets
    hosts allow = 192.168.1.0/24
```

3. [] [] [] [] [] []

```
sudo vi /etc/rsyncd.secrets
```

[] [] [] []

```
[ ] [ ] : [ ]
```

[] []

```
rsyncuser:mypassword123
```

[] [] [] [] [] []

```
sudo chmod 600 /etc/rsyncd.secrets
sudo chown root:root /etc/rsyncd.secrets
```

4. [] [] [] [] [] []

```
sudo mkdir -p /data/backup
sudo chown nobody:nobody /data/backup
```



rsyncd

1.

```
sudo systemctl enable rsyncd  
sudo systemctl start rsyncd
```

2.

```
sudo systemctl status rsyncd
```

3.

```
# 873  
sudo firewall-cmd --add-port=873/tcp --permanent  
sudo firewall-cmd --reload
```

4. SELinux

```
sudo setsebool -P rsync_full_access=1
```



1.

```
rsync -avz /local/path/ rsyncuser@server::backup
```

2.

```
rsync -avz rsyncuser@server::backup /local/path/
```

3.

~/rsync.pass

mypassword123

chmod 600 ~/rsync.pass

rsync -avz --password-file=~/rsync.pass /local/path/ rsyncuser@server::backup

1.

<div>path</div>	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>
<div>comment</div>	<div><div></div><div></div><div></div><div></div><div></div></div>
<div>read only</div>	<div><div></div><div></div><div></div> (yes/no)</div>
<div>list</div>	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div> (yes/no)</div>
<div>auth users</div>	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>
<div>secrets file</div>	<div><div></div><div></div><div></div><div></div><div></div></div>
<div>hosts allow</div>	<div><div></div><div></div><div></div><div></div><div></div><div></div> IP</div>
<div>hosts deny</div>	<div><div></div><div></div><div></div><div></div><div></div><div></div> IP</div>
<div>exclude</div>	<div><div></div><div></div><div></div><div></div> /<div></div><div></div></div>
<div>include</div>	<div><div></div><div></div><div></div><div></div> /<div></div><div></div></div>

2.

/etc/logrotate.d/rsyncd


```
/var/log/rsyncd.log {
    missingok
    notifempty
    sharedscripts
    delaycompress
    postrotate
        /bin/kill -HUP `cat /var/run/rsyncd.pid 2>/dev/null` 2>/dev/null || true
    endsript
}
```



1.

port = 8873



2. **IP**

hosts allow = 192.168.1.100, 10.0.0.0/24

3. **SSH**

rsync -avz -e "ssh -p 22" /local/path/ user@server:/remote/path/

4.

5.

sudo tail -f /var/log/rsyncd.log



1.

-
-
-

systemctl status rsyncd

ss -tulnp | grep rsync

2.

-
-
- SELinux

3. 1. 2.
- 3. 4. 5. 6. 7. 8. 9. 10.
 - 11. 12. 13. 14. 15. 16. 17. 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53. 54. 55. 56. 57. 58. 59. 60. 61. 62. 63. 64. 65. 66. 67. 68. 69. 70. 71. 72. 73. 74. 75. 76. 77. 78. 79. 80. 81. 82. 83. 84. 85. 86. 87. 88. 89. 90. 91. 92. 93. 94. 95. 96. 97. 98. 99. 100.

1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16. 17. 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53. 54. 55. 56. 57. 58. 59. 60. 61. 62. 63. 64. 65. 66. 67. 68. 69. 70. 71. 72. 73. 74. 75. 76. 77. 78. 79. 80. 81. 82. 83. 84. 85. 86. 87. 88. 89. 90. 91. 92. 93. 94. 95. 96. 97. 98. 99. 100.

Rocky Linux 8.5

rsync

Rocky Linux □ Docker □

YAPI □

YAPI □□□□□□□□□□□□□□□□

Docker □ YAPI □□□□□

Docker □□□□□□□□□□

Rocky Linux □□



1. □□□□

- Rocky Linux 8 □□□□
- □ 2GB □
- 10GB □□□□□

2. □□□□□□

```
sudo dnf install -y yum-utils device-mapper-persistent-data lvm2
```

□□□□ Docker

1. □ Docker CE □

```
sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

2. □ Docker □

```
sudo dnf install -y docker-ce docker-ce-cli containerd.io
```

3.

```
sudo systemctl enable --now docker
```

4.

```
sudo docker version
```



Docker Compose

1. Docker Compose

```
sudo curl -L "https://github.com/docker/compose/releases/download/v2.23.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

2.

```
sudo chmod +x /usr/local/bin/docker-compose
```

3.

```
docker-compose --version
```



YAPI

1.

```
mkdir ~/yapi && cd ~/yapi
```

2. `docker-compose.yml`

```
version: '3'

services:
  yapi-mongo:
    image: mongo:4
    container_name: yapi-mongo
    volumes:
      - ./mongo-data:/data/db
    restart: always
    environment:
      - MONGO_INITDB_ROOT_USERNAME=root
      - MONGO_INITDB_ROOT_PASSWORD=example
    ports:
      - "27017:27017"
    networks:
      - yapi-net

  yapi-web:
    image: jayfong/yapi:latest
    container_name: yapi-web
    depends_on:
      - yapi-mongo
    ports:
      - "3000:3000"
    environment:
      - YAPI_ADMIN_ACCOUNT=admin@yapi.com
      - YAPI_ADMIN_PASSWORD=admin123
      - YAPI_CLOSE_REGISTER=true
      - YAPI_DB_SERVERNAME=yapi-mongo
      - YAPI_DB_PORT=27017
      - YAPI_DB_DATABASE=yapi
      - YAPI_DB_USER=root
      - YAPI_DB_PASS=example
      - YAPI_DB_AUTH_SOURCE=admin
    restart: always
    networks:
      - yapi-net
```

```
networks:
  yapi-net:
    driver: bridge
```

3. 部署 YAPI 组件

```
docker-compose up -d
```



1. 部署数据库

部署数据库 2-3 个数据库

2. 部署 YAPI

部署 YAPI

```
http://IP:3000
```

3. 配置

- 配置邮箱 admin@yapi.com
- 配置密码 admin123



1. 部署数据库

部署数据库

2. 部署 YAPI

3. 备份

```
# 备份 MongoDB 数据
docker exec yapi-mongo sh -c 'exec mongodump -d yapi --archive' > yapi-backup-$(date +%Y%m%d).archive
```



1. 检查容器

```
docker-compose ps
```

2. 查看日志

```
docker-compose logs -f
```

3. 部署 YAPI

```
docker-compose pull yapi-web
docker-compose up -d
```

4. 清理环境

```
docker-compose down
```



1. 配置环境变量
2. 配置 Nginx 反向代理 HTTPS
3. 配置 MongoDB 数据库
4. 配置 IP 地址和 Nginx 端口

docker-compose.yml 文件

■■■■■■■■■

Rocky Linux ■■■■

YAPI ■■■■■■■■■■■■■■

YAPI

■■■■■■■■■■■■■■■■■■■■

SSH 连接

本地 SSH 连接

SSH (Secure Shell) 连接本地设备
rlogin 连接

Telnet 连接

连接

1. 本地连接

ssh username@hostname

连接

ssh user@example.com

2. 指定端口连接

ssh -p port_number username@hostname

连接

ssh -p 2222 user@example.com

连接

1. 本地 SSH 连接

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

- `-t` `rsa` (rsa, ed25519)
- `-b` `4096`
- `-C` `"your_email@example.com"`

2. `ssh-copy-id`

```
ssh-copy-id username@hostname
```

`ssh-copy-id` `~/.ssh/id_rsa.pub` `~/.ssh/authorized_keys`

`ssh-copy-id`

`ssh-copy-id` `~/.ssh/config`

```
Host myserver
  HostName example.com
  User username
  Port 2222
  IdentityFile ~/.ssh/id_rsa
```

`ssh-copy-id`

```
ssh myserver
```

`ssh-copy-id`

| <code>-v</code> | <code>ssh -v</code> (<code>ssh -vv, -vvv</code>) |
|-----------------|--|
| <code>-X</code> | <code>ssh -X</code> X11 |
| <code>-L</code> | <code>ssh -L</code> |
| <code>-R</code> | <code>ssh -R</code> |
| <code>-D</code> | <code>ssh -D</code> (SOCKS) |

| ssh | ssh |
|-----|-----|
| -N | ssh |
| -f | ssh |

ssh

1. ssh

```
ssh -L local_port:remote_host:remote_port username@hostname
```

2. ssh

```
ssh -R remote_port:local_host:local_port username@hostname
```

scp

1. scp

scp

```
scp local_file username@hostname:remote_path
```

scp

```
scp username@hostname:remote_file local_path
```

2. sftp

```
sftp username@hostname
```



1. 禁止 root 登录 (PermitRootLogin no in /etc/ssh/sshd_config)
2. 设置密码复杂度
3. 限制 SSH 连接数
4. 限制用户 (AllowUsers in sshd_config)
5. 安装 fail2ban 并配置



1. 检查 SSH 服务状态

systemctl status sshd

2. 查看 SSH 日志

journalctl -u sshd

3. 测试 SSH 连接

ssh -v username@hostname

cURL 入門

cURL (Client URL) 是透过命令行来操作 HTTP/HTTPS/FTP/SFTP 的客户端工具。API 也是透过 cURL 来操作。

1. GET 方法

GET 方法

```
curl https://example.com
```

- GET 方法
- GET 方法

```
curl "https://example.com/api?param1=value1&param2=value2"
```

POST 方法

```
curl -X POST https://example.com/api \
-d "key1=value1&key2=value2"
```

- JSON 数据

```
curl -X POST https://example.com/api \
-H "Content-Type: application/json" \
-d '{"key1": "value1", "key2": "value2"}'
```

2. 进阶



Header

```
curl -H "Authorization: Bearer token123" \  
-H "User-Agent: MyApp/1.0" \  
https://example.com/api
```

-  **Headers** 

```
curl -I https://example.com #  Headers
```



3. &



```
curl -X POST https://example.com/upload \  
-F "file=@/path/to/file.txt" \  
-F "name=myfile"
```

-   `multipart/form-data` 



```
curl -O https://example.com/file.zip #   
curl -o custom_name.zip https://example.com/file.zip # 
```

4. & Cookies

Basic Auth

```
curl -u username:password https://example.com
```

- `curl -u username https://example.com`

Cookie

```
curl --cookie "name=value" https://example.com
```

- `curl -c cookies.txt https://example.com/login`

- `curl -b cookies.txt https://example.com/dashboard`

5. &

```
curl -v https://example.com # & Headers
curl --trace-ascii debug.txt https://example.com #
```

```
curl -L https://example.com #
```

```
curl --limit-rate 100K -O https://example.com/largefile.zip
```

- `100K` = 100KB/s `M` (MB/s) `G` (GB/s)

6. `curl`

HTTP/HTTPS `curl`

```
curl -x http://proxy-server:8080 https://example.com
```

- `curl` `curl`

```
curl -x http://user:pass@proxy-server:8080 https://example.com
```

SOCKS5 `curl`

```
curl --socks5 127.0.0.1:1080 https://example.com
```

7. `curl`

Bash

```
response=$(curl -s https://example.com/api)
echo "$response"
```

- `-s` `curl`

8. `curl`

`curl` API `curl`

```
curl -s -o /dev/null -w "%{http_code}" https://example.com/api
```

- `-o /dev/null` `curl`
- `-w "%{http_code}"` `curl` HTTP `curl`



API



```
curl -w "DNS: %{time_namelookup} | Connect: %{time_connect} | Total: %{time_total}\n" \
-o /dev/null -s https://example.com
```



| Method | Example Command |
|----------|---|
| GET | curl https://example.com |
| POST | curl -X POST -d "data" https://example.com/api |
| JSON | curl -H "Content-Type: application/json" -d '{"key":"value"}' |
| Download | curl -O https://example.com/file.zip |
| Upload | curl -F "file=@localfile.txt" https://example.com/upload |
| Auth | curl -u user:pass https://example.com |
| Proxy | curl -x http://proxy:8080 https://example.com |
| Verbose | curl -v https://example.com |

Docker 与 ShowDoc

本文档将介绍如何在 Docker 中安装和配置 ShowDoc。

1. 环境准备

- 确保系统已安装 Docker 或 Docker Compose。
- 确保 80 端口未被占用。

2. 安装 ShowDoc

```
docker run -d --name showdoc \
-p 4999:80 \
-v /your/local/path:/var/www/html/ \
star7th/showdoc
```

注意：

- p 4999:80 表示将容器的 80 端口映射到主机的 4999 端口。
- v /your/local/path:/data/showdoc 表示将主机的 /your/local/path 目录挂载到容器的 /data/showdoc 目录。

3. 使用 Docker Compose 安装


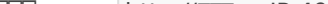
创建 docker-compose.yml 文件：

```
version: '3'
services:
  showdoc:
    image: star7th/showdoc
    container_name: showdoc
    ports:
      - "4999:80"
```

```
restart: unless-stopped
```

| | | | | |
|--|--|--|--|--|
| | | | | |
|--|--|--|--|--|

4. ShowDoc

-  `http://` `IP:4999`
- 

5. ☐ ☐ HTTPS☐☐☐

```
server {  
    listen 443 ssl;  
    server_name doc.yourdomain.com;  
    ssl_certificate /path/to/cert.pem;  
    ssl_certificate_key /path/to/key.pem;  
  
    location / {  
        proxy_pass http://localhost:4999;  
        proxy_set_header Host $host;  
        proxy_set_header X-Real-IP $remote_addr;  
    }  
}
```

6.

| | | | |
|--|--|--|--|
| | | | |
|--|--|--|--|

-   `/your/local/path` 
-  



- 1. 名称 端口 -p 容器端口 主机端口
- 2. 名称 容器端口

```
chmod -R 777 /your/local/path
```

- 3. 名称 命令

```
docker-compose pull && docker-compose up -d
```



- 名称 Docker 名称 ShowDoc 名称 1 名称 容器端口
- 名称 volumes 容器端口
- 容器端口 Nginx 名称 HTTPS 容器端口

容器端口 MySQL容器端口 ShowDoc 名称 名称