

Day 26 - Remotely Turning on PCs With Wake-On-Lan

What is Wake-on-LAN? [🔗](#)

Wake-on-LAN (WOL) is a hardware and software feature that allows a powered-off or sleeping computer to be turned on remotely over a Local Area Network (LAN). The mechanism works by sending a specially crafted network packet, known as a **magic packet**, to the target device's **MAC address**.

What is a Magic Packet? [🔗](#)

A magic packet is a broadcast frame containing:

- 6 bytes of `FF` (255)
- Followed by the MAC address of the device repeated 16 times (16 × 6 bytes)

The system's network interface card (NIC), if configured to do so, listens for such packets even when the system is in a low-power state and will trigger a wake-up if a matching packet is received.

How to Check if Wake-on-LAN is Supported and Enabled [🔗](#)

Check if NIC supports WOL [🔗](#)

```
1 sudo ethtool <interface>
```

Example:

```
1 sudo ethtool eno1
```

Look for the line:

```
1 Supports Wake-on: pumbg
2 Wake-on: g
```

- `g` means WOL via magic packet is supported.

```
[root@Rockylinux1 ~]# ethtool eno1
Settings for eno1:
  Supported ports: [ TP ]
  Supported link modes:  10baseT/Half 10baseT/Full
                        100baseT/Half 100baseT/Full
                        1000baseT/Full
  Supported pause frame use: Symmetric Receive-only
  Supports auto-negotiation: Yes
  Supported FEC modes: Not reported
  Advertised link modes:  10baseT/Half 10baseT/Full
                        100baseT/Half 100baseT/Full
                        1000baseT/Full
  Advertised pause frame use: Symmetric Receive-only
  Advertised auto-negotiation: Yes
  Advertised FEC modes: Not reported
  Link partner advertised link modes:  10baseT/Half 10baseT/Full
                                      100baseT/Half 100baseT/Full
  Link partner advertised pause frame use: No
  Link partner advertised auto-negotiation: Yes
  Link partner advertised FEC modes: Not reported
  Speed: 100Mb/s
  Duplex: Full
  Auto-negotiation: on
  Port: Twisted Pair
  PHYAD: 1
  Transceiver: internal
  MDI-X: on (auto)
  Supports Wake-on: pumbg
  Wake-on: g
  Current message level: 0x00000007 (7)
                        drv probe link
  Link detected: yes
```

Wake-on-lan is enabled

- If it's `d`, it's disabled. You can enable it as shown below.

```
1 sudo ethtool -s eth0 wol g
```

Sending a Magic Packet (Linux) [↗](#)

Ubuntu/Debian [↗](#)

```
1 sudo apt install wakeonlan
2 wakeonlan <mac-address>
```

RHEL/Rocky Linux [↗](#)

```
1 sudo dnf install perl-Net-Wake
2 wakeonlan <mac-address>
```

Python Script to Send Magic Packet [↗](#)

You can also use a simple Python script to send a magic packet.

Script: `arise.py` [↗](#)

```
1 #!/usr/bin/env python3
2
3 import socket
4 import sys
5
6 MACS = {
7     "rocky1": "00:11:22:33:44:55"
8 }
9
10 def wake_on_lan(mac):
11     mac_bytes = bytes.fromhex(mac.replace(':', '').replace('-', ''))
12     magic_packet = b'\xff' * 6 + mac_bytes * 16
13     with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
14         s.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
15         s.sendto(magic_packet, ('<broadcast>', 9))
16
17 if __name__ == "__main__":
18     if len(sys.argv) < 2:
19         print("Usage: arise <device-name>")
20         sys.exit(1)
21
22     target = sys.argv[1]
23     if target not in MACS:
24         print(f"Unknown device '{target}'. Available: {'', '.join(MACS)}")
25         sys.exit(1)
26
27     wake_on_lan(MACS[target])
```

Installation and Usage [↗](#)

1. Save the script as `arise.py`
2. Make it executable:

```
1 chmod +x arise.py
```

3. Move it to your PATH:

```
1 sudo mv arise.py /usr/local/bin/arise
```

Now you can wake a machine using:

```
1 arise rocky1
```

Sample Working [↗](#)

On the target PC, issue a shutdown command using:

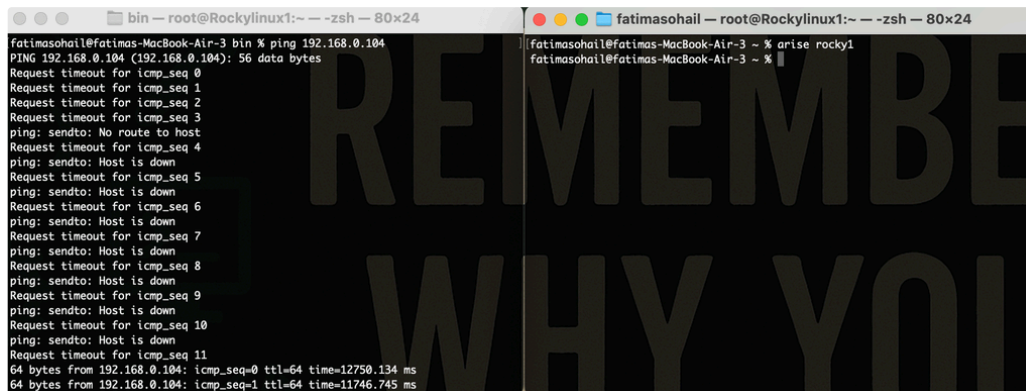
```
1 systemctl poweroff --now
```

```
[root@Rockylinux1 ~]# systemctl poweroff --now
[root@Rockylinux1 ~]# Connection to 192.168.0.104 closed by remote host.
Connection to 192.168.0.104 closed.
fatimasohail@fatimas-MacBook-Air-3 bin %
```

The system is no longer accessible, as verified by ping.

```
1 ping 192.168.0.104
```

Then run the script, and after a few seconds, the host will be reachable.



The screenshot shows two terminal windows side-by-side. The left window, titled 'bin — root@Rockylinux1:~ — zsh — 80x24', shows a ping command being executed from a Mac: 'fatimasohail@fatimas-MacBook-Air-3 bin % ping 192.168.0.104'. The output shows 11 failed ping requests with 'Request timeout for icmp_seq' and 'ping: sendto: Host is down' messages. The right window, titled 'fatimasohail — root@Rockylinux1:~ — zsh — 80x24', shows the execution of the 'arise' script: 'fatimasohail@fatimas-MacBook-Air-3 ~ % arise rocky1'. The output shows the script successfully connecting to the host.

```
fatimasohail@fatimas-MacBook-Air-3 bin % ping 192.168.0.104
PING 192.168.0.104 (192.168.0.104): 56 data bytes
Request timeout for icmp_seq 0
Request timeout for icmp_seq 1
Request timeout for icmp_seq 2
Request timeout for icmp_seq 3
ping: sendto: No route to host
Request timeout for icmp_seq 4
ping: sendto: Host is down
Request timeout for icmp_seq 5
ping: sendto: Host is down
Request timeout for icmp_seq 6
ping: sendto: Host is down
Request timeout for icmp_seq 7
ping: sendto: Host is down
Request timeout for icmp_seq 8
ping: sendto: Host is down
Request timeout for icmp_seq 9
ping: sendto: Host is down
Request timeout for icmp_seq 10
ping: sendto: Host is down
Request timeout for icmp_seq 11
64 bytes from 192.168.0.104: icmp_seq=0 ttl=64 time=12750.134 ms
64 bytes from 192.168.0.104: icmp_seq=1 ttl=64 time=11746.745 ms

fatimasohail@fatimas-MacBook-Air-3 ~ % arise rocky1
fatimasohail@fatimas-MacBook-Air-3 ~ %
```