

# Uptime Kuma

All docs related to Uptime Kuma

- [Uptime Kuma Installation](#)
  - [\[How-To\] Install Uptime Kuma on Debian 12 LXC](#)
- [Uptime Kuma Configuration](#)
  - [\[How-To\] Use Uptime Kuma Behind Reverse Proxy](#)
  - [\[How-To\] Reset Uptime Kuma Password via CLI](#)

# Uptime Kuma Installation

All docs related to Uptime Kuma Installation

# [How-To] Install Uptime Kuma on Debian 12 LXC

## Purpose

The purpose of this document is to show how to install uptime kuma on a Debian 12 LXC in proxmox.

## Prerequisites

List of prerequisites:

- Root user or sudo user
- Debian 12 LXC

## Installation Instructions - Docker

Using Docker deploy requires you to install docker and docker-compose on the LXC before proceeding.

### Step 1: Install Uptime Kuma

Run the following line to install uptime kuma via docker:

```
docker run -d --restart=always -p 3001:3001 -v uptime-kuma:/app/data --name uptime-kuma louislam/uptime-kuma:1
```

Thats it! Uptime Kuma is now running on [HTTP://localhost:3001](http://localhost:3001)

Filesystem support for POSIX file locks is required to avoid SQLite database corruption. Be aware of possible file locking problems such as those commonly encountered with NFS.  
**Please map the `/app/data` -folder to a local directory or volume.**

Browse to [HTTP://localhost:3001](http://localhost:3001)

## Step 2: Change Port or Volume (Optional)

Run the following line to adjust the port or volume and replace YOUR\_PORT and YOU\_DIR OR VOLUME with your information

```
docker run -d --restart=always -p <YOUR_PORT>:3001 -v <YOUR_DIR OR VOLUME>:/app/data --name uptime-kuma louislam/uptime-kuma:1
```

Thats it! Uptime Kuma is now running on [HTTP://localhost:3001](http://localhost:3001)

# Installation Instructions - Non-Docker

## Step 1: Prerequisites

Ensure you have the non-docker prerequisite completed:

- Node.js 14/16/18/20.4
- npm 9
- GIT
- pm2

If you don't use the respective installer line below to get them installed:

### Node.js 14/16/18/20.4

```
scripts
```

### Npm 9

```
scripts
```

### Git

```
sudo apt install git -y
```

### Pm2

```
npm install pm2 -g && pm2 install pm2-logrotate
```

## Step 2: Install Uptime Kuma

Run the following script to verify you version of npm is at the correct version needed:

```
npm install npm@9 -g
```

Run the following to clone the repo for Uptime Kuma:

```
git clone https://github.com/louislam/uptime-kuma.git
```

Run the following script to change directories into the uptime-kuma folder downloaded and use npm to run setup:

```
cd uptime-kuma
```

```
npm run setup
```

## Step 3: Start the Service

Option 1 to start the service:

```
node server/server.js
```

Option 2 to start the service (Recommended) Running it in the background using PM2:

```
pm2 start server/server.js --name uptime-kuma
```

Thats it! Uptime Kuma is now running on [HTTP://localhost:3001](http://localhost:3001)

## Useful PM2 Commands

Here are some useful PM2 Commands:

- If you want to see the current console output

```
pm2 monit
```

- If you want to add it to startup

```
pm2 save && pm2 startup
```

<https://github.com/louislam/uptime-kuma/wiki/%F0%9F%94%A7-How-to-Install>

# Uptime Kuma Configuration

All docs related to Uptime Kuma configuration

# [How-To] Use Uptime Kuma Behind Reverse Proxy

## Purpose

This document aims to show how to configure your reverse proxy configuration for Uptime Kuma as it is a web socket app.

## Prerequisites

List of prerequisites:

- Root user or sudo user
- Uptime Kuma Server

## Reverse Proxy Configuration

### Nginx Reverse Proxy:

For Nginx with SSL:

```
server {  
    listen 443 ssl http2;  
    # Remove '#' in the next line to enable IPv6  
    # listen [::]:443 ssl http2;  
    server_name sub.domain.com;  
    ssl_certificate /path/to/ssl/cert/crt;  
    ssl_certificate_key /path/to/ssl/key/key;  
    # *See "With SSL (Certbot)" below for details on automating ssl certificates  
  
    location / {  
        proxy_set_header X-Real-IP $remote_addr;  
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
```

```
proxy_set_header Host $host;
proxy_pass      http://localhost:3001/;
proxy_http_version 1.1;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "upgrade";
}
}
```

### For Nginx with SSL (Certbot):

```
server {
    # If you don't have one yet, you can set up a subdomain with your domain registrar (e.g. Namecheap)
    # Just create a new host record with type='A Record', host='<subdomain>', value='<ip_address>'.

    server_name your_subdomain.your_domain.your_tld;

    location / {
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $host;
        proxy_pass      http://localhost:3001/;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }
}

# Once that's completed, you can run
# sudo apt install python3-certbot-nginx
# sudo certbot --nginx -d your_domain -d your_subdomain.your_domain -d www.your_domain
# And Certbot will auto-populate this nginx .conf file for you, while also renewing your certificates automatically
in the future.
```

### For Nginx without SSL:

```
server {
    listen 80;
    # Remove '#' in the next line to enable IPv6
```



```
# listen [::]:80;

server_name sub.domain.com;

location / {
    proxy_pass http://localhost:3001;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_set_header Host $host;
}

}
```

## Apache Reverse Proxy:

For Apache With SSL:

```
<VirtualHost *:443>
    ServerName sub.domain.com
    SSLEngine On
    SSLCertificateFile /path/to/ssl/cert/crt
    SSLCertificateKeyFile /path/to/ssl/key/key
    # Protocol 'h2' is only supported on Apache 2.4.17 or newer.
    Protocols h2 http/1.1
    ProxyPreserveHost on
    ProxyPass / http://localhost:3001/
    RewriteEngine on
    RewriteCond %{HTTP:Upgrade} =websocket
    RewriteRule /(.*) ws://localhost:3001/$1 [P,L]
    RewriteCond %{HTTP:Upgrade} !=websocket
    RewriteRule /(.*) http://localhost:3001/$1 [P,L]
</VirtualHost>
```

For Apache Without SSL:

```
<VirtualHost *:80>
    ServerName sub.domain.com
    ProxyPreserveHost on
    ProxyPass / http://localhost:3001/
    RewriteEngine on
    RewriteCond %{HTTP:Upgrade} websocket [NC]
    RewriteCond %{HTTP:Connection} upgrade [NC]
```

```
RewriteRule ^/?(.*) "ws://localhost:3001/$1" [P,L]
</VirtualHost>
```

## Caddy Reverse Proxy:

Caddy Normal:

```
subdomain.domain.com {
    reverse_proxy 127.0.0.1:3001
}
```

Caddy with Docker-Compose:

```
version: '3'
networks:
  default:
    name: 'proxy_network'
services:
  uptime-kuma:
    image: louislam/uptime-kuma:1
    restart: unless-stopped
    volumes:
      - /srv/uptime:/app/data
  labels:
    caddy: status.example.org
    caddy.reverse_proxy: "*" {{upstreams 3001}}"
  caddy:
    image: "lucaslorentz/caddy-docker-proxy:ci-alpine"
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - /srv/caddy/:/data
    restart: unless-stopped
    environment:
      - CADDY_INGRESS_NETWORKS=proxy_network
```

## HTTPS-Portal Reverse Proxy:

## Https Normal:

```
version: '3.3'

services:
  https-portal:
    image: steveltn/https-portal:1
    ports:
      - '80:80'
      - '443:443'
    links:
      - uptime-kuma
    restart: always
    environment:
      DOMAINS: 'status.domain.com -> http://uptime-kuma:3001'
      STAGE: 'production' # Don't use production until staging works
      # FORCE_RENEW: 'true'
      WEBSOCKET: 'true'
    volumes:
      - https-portal-data:/var/lib/https-portal

  uptime-kuma:
    image: louislam/uptime-kuma:1
    container_name: uptime-kuma
    volumes:
      - ./uptime-kuma:/app/data
    ports:
      - 3001:3001

volumes:
  https-portal-data:
```

## HAProxy:

No special configuration is required when using HAProxy as a reverse proxy although you may wish to add the `timeout tunnel` option to either the `defaults`, `listen`, or `backend` sections. If using the `timeout tunnel` option, it is also recommended to set `timeout client-fin` to handle instances where the client stops responding.

Read more: <http://cbonte.github.io/haproxy-dconv/2.4/configuration.html#4.2-timeout%20tunnel>

<https://github.com/louislam/uptime-kuma/wiki/Reverse-Proxy#apache>

# [How-To] Reset Uptime Kuma Password via CLI

## Purpose

This document aims to show how to reset the password for your UI user for uptime kuma via the CLI of the server it is running on.

## Prerequisites

List of prerequisites:

- Root user or sudo user
- Server running uptime Kuma

## Reset Instructions

### Step 1

Login with the root user, or login with normal user and use the follow to elevate to root:

```
sudo su -l
```

### Step 2

Run the following command to enter the Uptime Kuma docker container CLI:

```
docker exec -it uptime-kuma bash
```

### Step 3

Run the following command to access the reset tool:

```
npm run reset-password
```

Then, follow the prompts to reset and test in the UI of Uptime Kuma.