How to Replace a String in Multiple Files in a Linux Folder?

written by sysadmin | 17 January 2026

I want to replace a string that is in multiple files in a Linux folder.

Problem

How to replace a string in multiple files in a Linux folder?

Solution

For example, I want to search for the database.php word scattered across multiple PHP files in a folder, and I use the Linux command below:

```
grep -R database.php *
```

```
root@docker:/var/www# ls
portal  portal-ori
root@docker:/var/www#
root@docker:/var/www# grep -R database.php *
portal/app1/index.php:require_once '../includes/database.php';
portal/app1/create.php:require_once '../includes/database.php';
portal/app1/delete.php:require_once '../includes/database.php';
portal/app1/edit.php:require_once '../includes/database.php';
portal/reset_password.php:require_once 'includes/database.php';
portal/app2/index.php:require_once '../includes/database.php';
portal/app2/create.php:require_once '../includes/database.php';
portal/app2/delete.php:require_once '../includes/database.php';
portal/app2/edit.php:require_once '../includes/database.php';
portal/register.php:require_once 'includes/database.php';
portal/login.php:require_once 'includes/database.php';
portal/run_scripts/add.php:                                placeholder="e.g., backup_database.php"
portal/create_admin.php:require_once 'includes/database.php';
portal-ori/app1/index.php:require_once '../includes/database.php';
portal-ori/app1/create.php:require_once '../includes/database.php';
portal-ori/app1/delete.php:require_once '../includes/database.php';
portal-ori/app1/edit.php:require_once '../includes/database.php';
portal-ori/reset_password.php:require_once 'includes/database.php';
portal-ori/app2/index.php:require_once '../includes/database.php';
portal-ori/app2/create.php:require_once '../includes/database.php';
portal-ori/app2/delete.php:require_once '../includes/database.php';
portal-ori/app2/edit.php:require_once '../includes/database.php';
portal-ori/register.php:require_once 'includes/database.php';
portal-ori/login.php:require_once 'includes/database.php';
portal-ori/create_admin.php:require_once 'includes/database.php';
root@docker:/var/www#
```

Find the string in multiple folders

In the image above, the word is scattered across multiple Linux files and in different Linux folders. So that you don't go into the file that is in a different folder one by one and then change the word database.php, for example, to config.php manually, but you can run the command below:

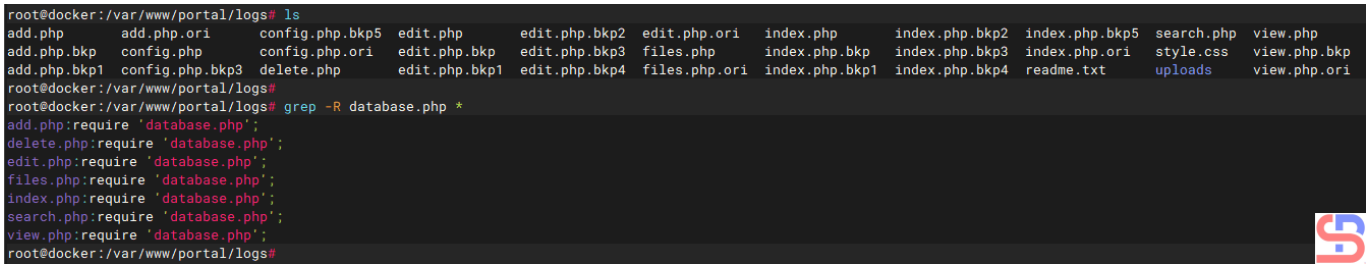
```
find . -type f -name "*.php" -exec sed -i 's/database\.php/config.php/g' {} +
```

which results in the image below:



Replace the string in multiple folders

In the image above, you can see that the word has been successfully changed to config.php, which is not found when you search for database.php. If you just change the database.php word to config.php in a folder, as shown in the image below:



Find the string in a folder

You can use the format below to replace the word:

```
sed -i 's/old_word/new_word/g' filename
```

So you can run the command below to change the word database.conf to config.php

```
sed -i 's/database\.php/config.php/g' *.php
```

So it will look like the image below:

```
root@docker:/var/www/portal/logs# grep -R database.php *
add.php:require 'database.php';
delete.php:require 'database.php';
edit.php:require 'database.php';
files.php:require 'database.php';
index.php:require 'database.php';
search.php:require 'database.php';
view.php:require 'database.php';
root@docker:/var/www/portal/logs#
root@docker:/var/www/portal/logs# sed -i 's/database\.php/config.php/g' *.php
root@docker:/var/www/portal/logs#
root@docker:/var/www/portal/logs# grep -R database.php *
root@docker:/var/www/portal/logs#
```

Replace the string in a folder

In the image above, you can see that the string database.php has been changed to config.php.

Note

If you're still unsure about running the two commands above to change a word scattered across multiple files, you can preview the results you expect. For example, if you want to preview the results first before you permanently change the changes in a folder, then you can run the command below:

```
sed 's/database\.php/config.php/g' *.php | grep database.php
```

and the result will be as shown in the image below:

```
root@docker:/var/www/portal/logs# grep -R database.php *
add.php:require 'database.php';
delete.php:require 'database.php';
edit.php:require 'database.php';
files.php:require 'database.php';
index.php:require 'database.php';
search.php:require 'database.php';
view.php:require 'database.php';
root@docker:/var/www/portal/logs#
root@docker:/var/www/portal/logs# sed 's/database\.php/config.php/g' *.php | grep config.php
require 'config.php';
require 'config.php';
require 'config.php';
require 'config.php';
require 'config.php';
require 'config.php';
require 'config.php';
```



Preview the result

And if you want to automatically back up when changing a string in multiple files, then use the command below:

```
sed -i.bak 's/database\.php/config.php/g' *.php
```

Then the result will be as shown in the image below:

```
root@docker:/var/www/portal/logs# grep -R database.php *
add.php:require 'database.php';
delete.php:require 'database.php';
edit.php:require 'database.php';
files.php:require 'database.php';
index.php:require 'database.php';
search.php:require 'database.php';
view.php:require 'database.php';
root@docker:/var/www/portal/logs#
root@docker:/var/www/portal/logs# sed -i.bak 's/database\.php/config.php/g' *.php
root@docker:/var/www/portal/logs#
root@docker:/var/www/portal/logs# grep -R database.php *
add.php.bak:require 'database.php';
delete.php.bak:require 'database.php';
edit.php.bak:require 'database.php';
files.php.bak:require 'database.php';
index.php.bak:require 'database.php';
search.php.bak:require 'database.php';
view.php.bak:require 'database.php';
root@docker:/var/www/portal/logs#
```



Back up the file(s) automatically

In the image, you can see that files that have the word database.php are automatically backed up to files with the

.bak extension.

References

unix.stackexchange.com
askubuntu.com
stackoverflow.com

How to Install Uptime Kuma Application on Ubuntu?

written by sysadmin | 17 January 2026

The previous articles explained how to install the uptime kuma application on Docker, either using the SQLite database or using the MariaDB database on Docker or using the MariaDB database on the host. This article will explain how to install the uptime kuma application without using Docker but using packages.

Problem

How to install uptime kuma application on Ubuntu?

Solution

Here are the steps to install uptime kuma application on Ubuntu:

1. Install the packages

Run the commands below to install the required packages:

```
sudo apt update -y
sudo apt install nginx mariadb-server git -y
```

Then, install nodejs using the command below:

```
curl -fsSL https://deb.nodesource.com/setup_lts.x | sudo -E bash - && sudo apt install -y nodejs
```

After that, download the uptime kuma application by running the command below:

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

Next, copy the commands below to install the uptime kuma application:

```
sudo npm run setup
sudo npm install pm2 -g
sudo pm2 install pm2-logrotate
sudo pm2 start server/server.js --name uptime-kuma
sudo pm2 startup
```

2. Configure MariaDB

Access MariaDB and run the queries below:

```
Akses ke MariaDB dan jalankan query-query di Bawah ini:
CREATE DATABASE uptime_kuma;
CREATE USER 'kuma-user'@'%' IDENTIFIED BY 'kumapass123';
GRANT ALL PRIVILEGES ON uptime_kuma.* TO 'kuma-user'@'%';
FLUSH PRIVILEGES;
\q
```

3. Configure web server

If you use Apache, create a file at **/etc/apache2/sites-available/kuma.conf** and copy the script below to the file:

```
<VirtualHost *:80>
ServerName yourdomain.com
DocumentRoot /var/www/html/

ProxyPass / http://localhost:3001/
RewriteEngine on
RewriteCond %{HTTP:Upgrade} websocket [NC]
```

```
RewriteCond %{HTTP:Connection} upgrade [NC]
RewriteRule ^/?(.*) "ws://localhost:3001/$1" [P,L]

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

</VirtualHost>
```

then run the command below:

```
sudo a2enmod rewrite
sudo a2enmod proxy
sudo a2enmod proxy_http
sudo a2ensite kuma.conf
```

Check if there is an error in Apache and if there is no error, reload Apache using the command below:

```
apachectl -t
sudo systemctl reload apache2
```

INFO

If your server is running an nginx webserver, then in the file **/etc/nginx/conf.d/uptime-kuma.conf** insert the script below:

```
server {
    listen 80;
    server_name uptime-kuma.yourdomainname.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;
        proxy_set_header   X-Real-IP $remote_addr;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header   X-Forwarded-Proto $scheme;

        # Added WebSocket support
        proxy_set_header   Sec-WebSocket-Key $http_sec_websocket_key;
        proxy_set_header   Sec-WebSocket-Version $http_sec_websocket_version;
        proxy_set_header   Sec-WebSocket-Extensions
$http_sec_websocket_extensions;
```

```
        # Improve performance of this reverse proxy
        proxy_buffering    off;
    }

    # Redirect HTTP to HTTPS if needed for encryption
    # Uncomment the following lines if you have SSL enabled
    # return 301 https://$host$request_uri;
}
```

Use the command below to check if there is an error in the nginx configuration and then reload nginx:

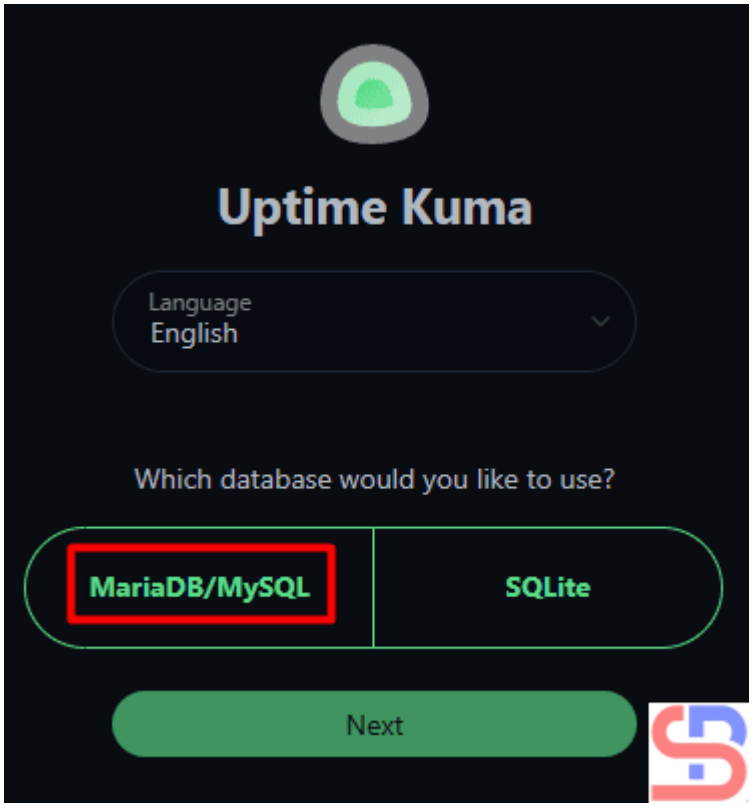
```
sudo nginx -t
sudo systemctl reload nginx
```

4. Access uptime kuma

Open your browser, and type:

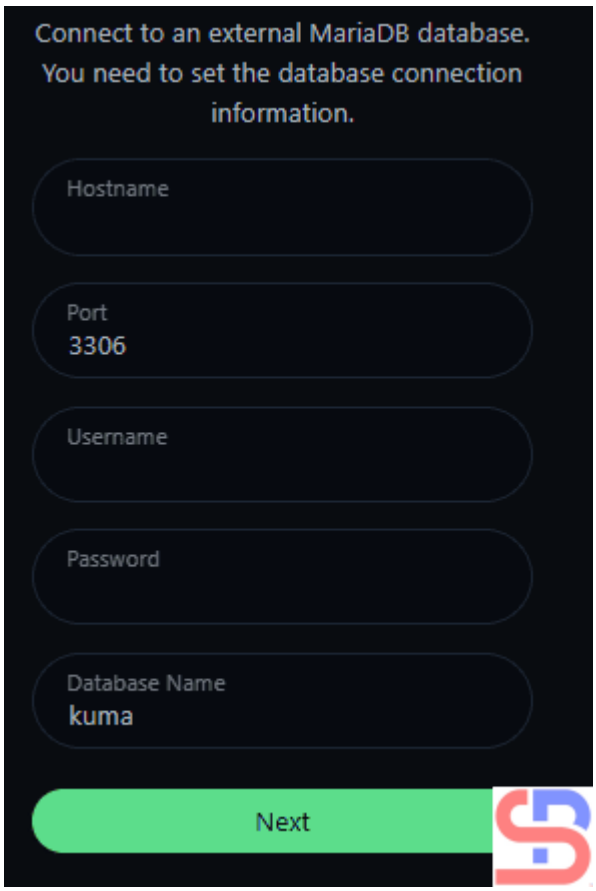
```
http://ip_server:3001
```

then there will be a display like below:



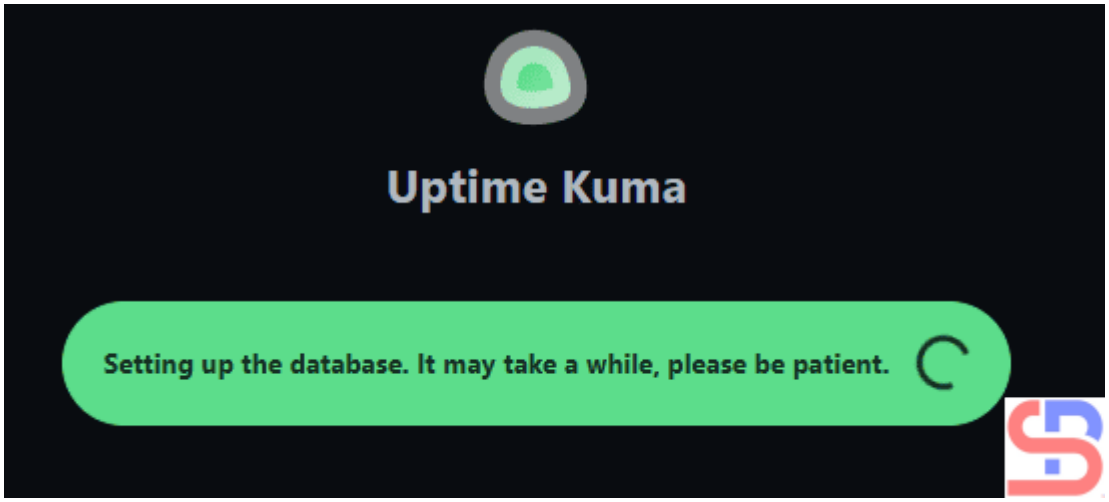
Click the MariaDB/MySQL button

Click **MariaDB/MySQL**, your screen will appear similar to the picture below:



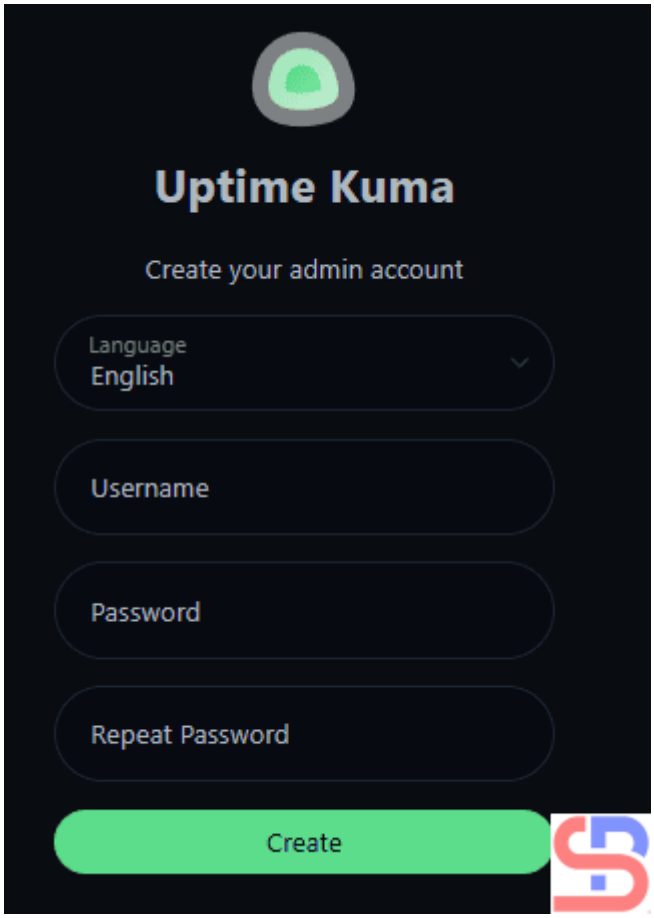
Fill in the columns for the database

Enter in the columns above the values □□that correspond to the query commands. Click the **Next** button, your screen will show up similar to the one below:



Setting up the database

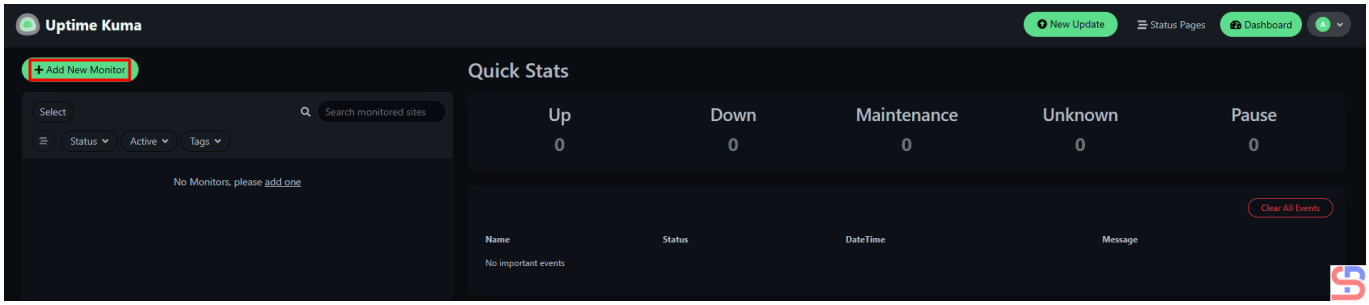
You have to wait until finish, and after that, your screen will appear similar to the image shown below:



Fill in the columns for the admin account

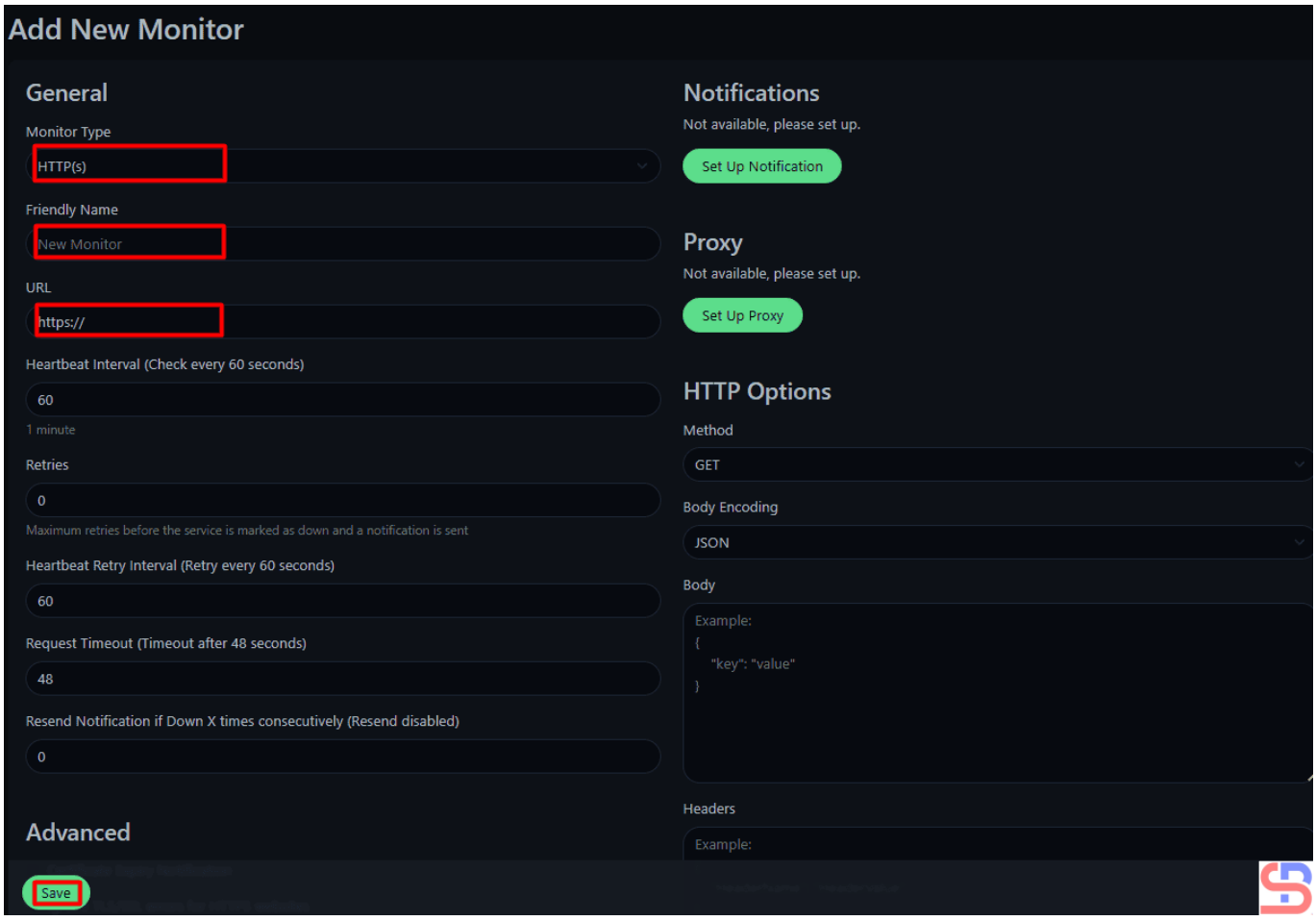
Enter in the columns above the value you want and press the **Create** button then a display will appear similar to the

image provided below:



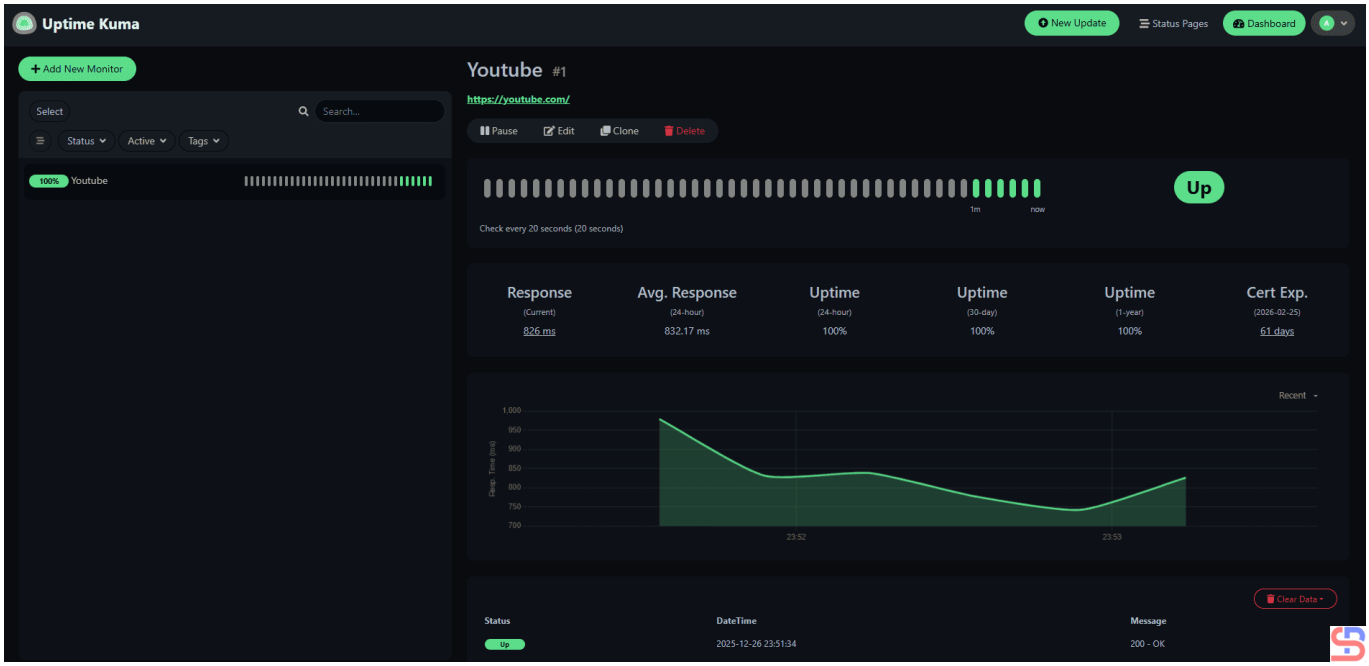
Display of uptime kuma application

If you want to monitor the website, click the **Add New Monitor** button at the top left of the site , an image similar to the one shown will appear:



Create a new host or a website to monitor in uptime kuma

Fill in the required fields (at least fill in the **Monitor Type**, **Friendly Name**, and **URL** columns) and press the **Save** button, then the host you have filled in will look like in the image below:



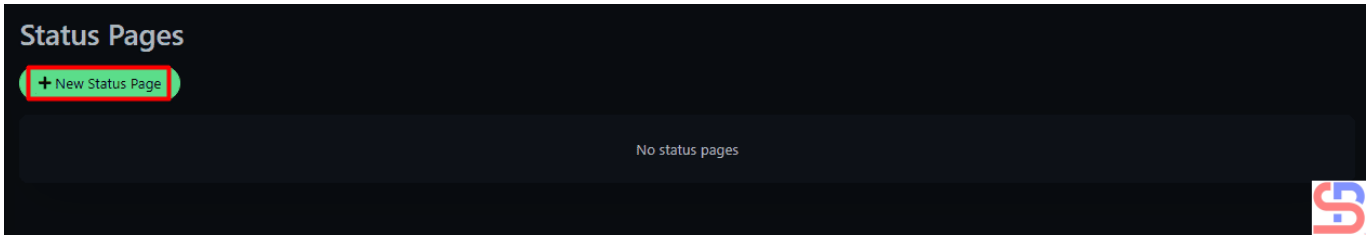
Monitor the host

If you just want to display the status without displaying many attributes then you can click the **Status Pages** button at the top right of the site like the image below:



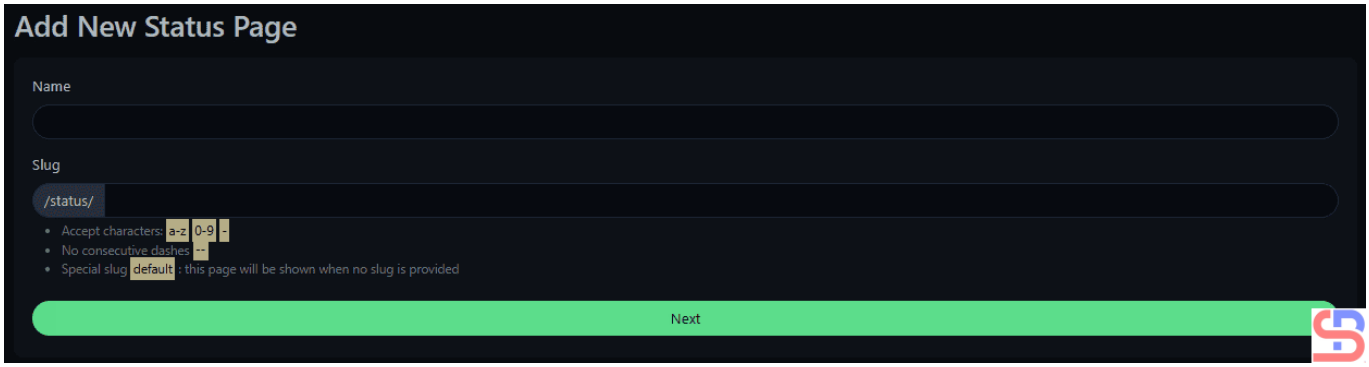
Click the Status Pages button

After you press the Status Pages button, the following image will appear:



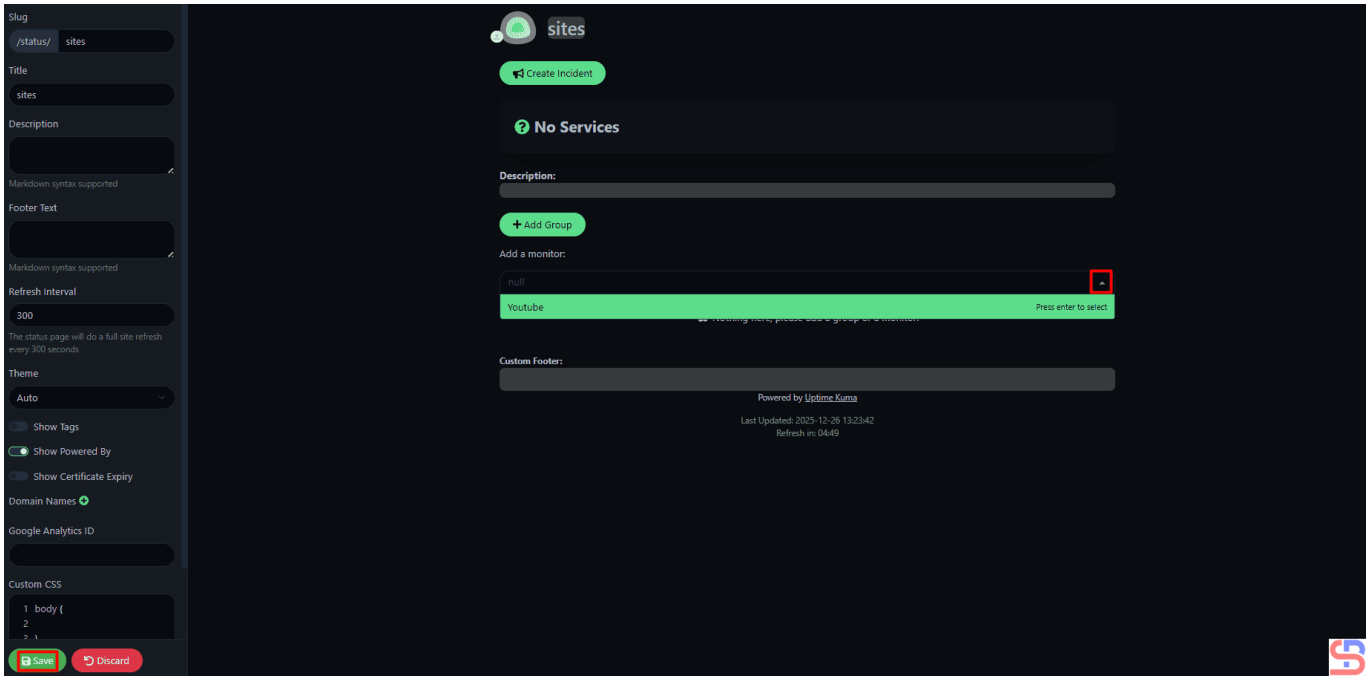
Create the Status Page page

Click the **New Status Page** button, and an image will appear similar to the one shown below:



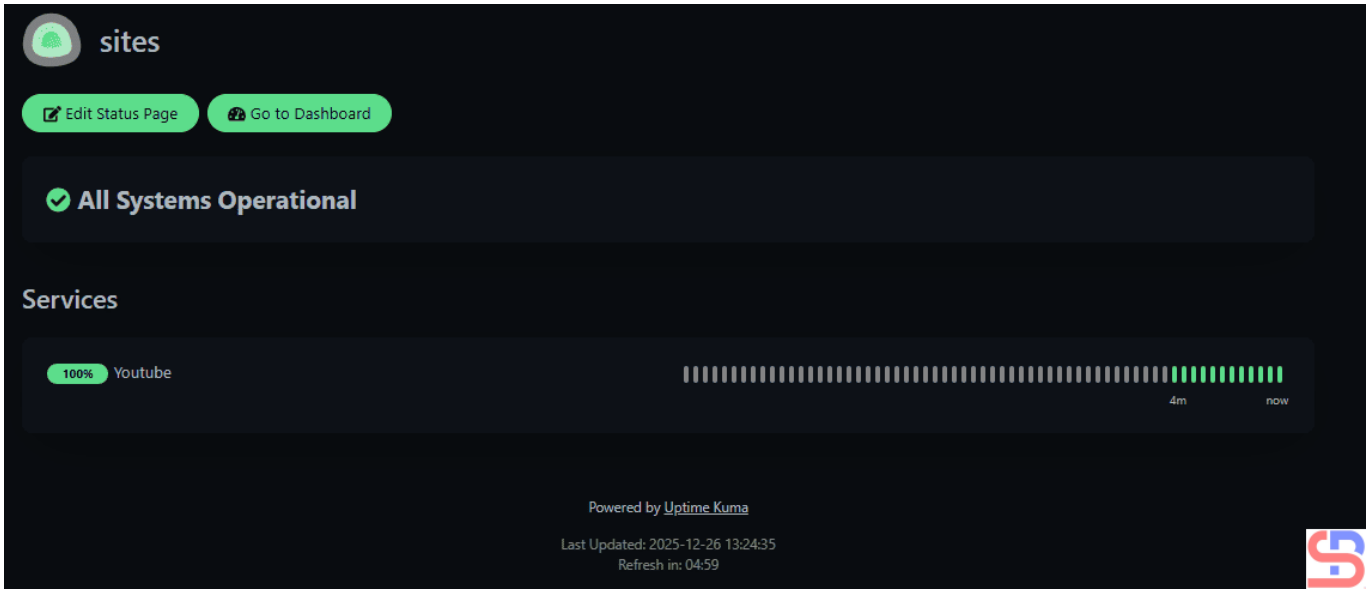
Create the Status Page page

Enter the name and slug you want (I wrote the sites for the name and slug), then press the Next button, then there will be a display as below:



Insert the host or the monitor in the Status Page

Enter the host you want to display on the Status Page, after that click the Save button, then there will be a display as below:



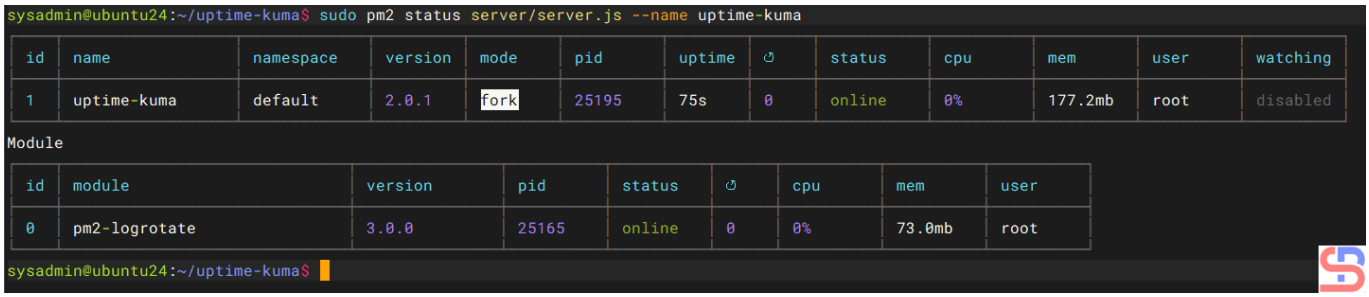
Display of Status Page

You can see that the hosts to be monitored look simpler and you can give the URL to other parties to also monitor these hosts.

Note

If you want to check the status of uptime kuma in the server, run the command below :

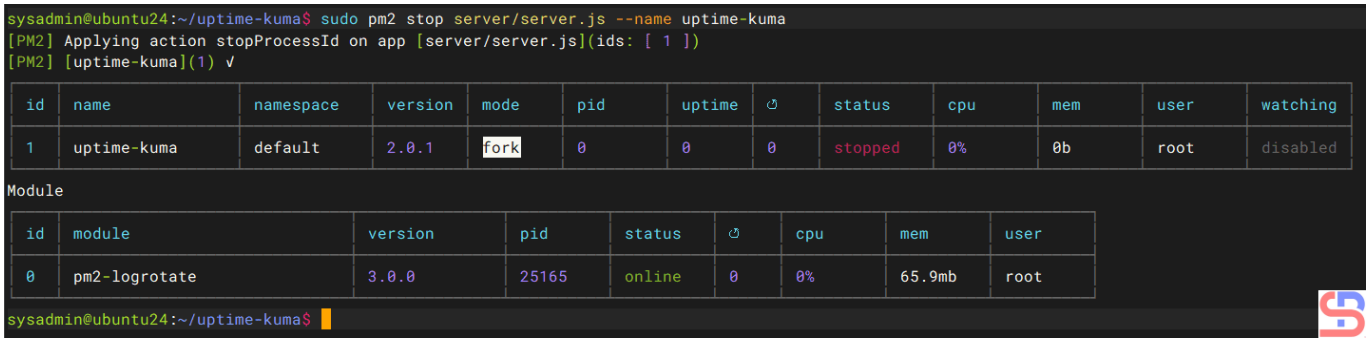
```
sudo pm2 status server/server.js --name uptime-kuma
```



Check the status of uptime kuma

But if you want stop uptime kuma in the server, run the command below :

```
sudo pm2 stop server/server.js --name uptime-kuma
```



Stop status kuma service

References

uptimekuma.org
hostmycode.in
youtube.com

How to Install Uptime Kuma in Docker with MariaDB on the Host?

written by sysadmin | 17 January 2026

The previous article explained how to install the Kuma uptime application using Docker and using the MariaDB database which also runs on Docker. This article will explain about installing the Kuma uptime application using Docker but using MariaDB on the host.

Problem

How to install uptime kuma in docker with MariaDB on the host?

Solution

Here are the steps to install the kuma uptime application using docker but using MariaDB on the host:

1. Configure MariaDB

Make sure you have installed the MariaDB database on your server and then change the file **/etc/mysql/mariadb.conf.d/50-server.cnf** in the **bind-address** section to be as below:

```
bind-address            = 0.0.0.0
```

Next, restart MariaDB using the command:

```
sudo systemctl restart mariadb
```

After that, access to MariaDB and run the command below:

```
CREATE DATABASE uptime_kuma;
CREATE USER 'kuma-user'@'%' IDENTIFIED BY 'kumapass123';
GRANT ALL PRIVILEGES ON uptime_kuma.* TO 'kuma-user'@'%';
FLUSH PRIVILEGES;
\q
```

2. Create a docker compose file

Create a compose folder in the /opt folder using the command below:

```
sudo mkdir -p /opt/compose/uptime-kuma/
cd /opt/compose/uptime-kuma/
```

After that, create a **docker-compose.yaml** file and copy the script below:

```
services:
  uptime-kuma:
    image: louislam/uptime-kuma:2
    container_name: uptime-kuma
    restart: unless-stopped
    ports:
      - "3001:3001"
    extra_hosts:
      - "host.docker.internal:host-gateway"
    environment:
```

```
      UPTIME_KUMA_DB_TYPE: mariadb
      UPTIME_KUMA_DB_HOSTNAME: host.docker.internal
      UPTIME_KUMA_DB_PORT: 3306
      UPTIME_KUMA_DB_NAME: ${MARIADB_DATABASE}
      UPTIME_KUMA_DB_USERNAME: ${MARIADB_USER}
      UPTIME_KUMA_DB_PASSWORD: ${MARIADB_PASSWORD}
    volumes:
      - kuma-data:/app/data

volumes:
  kuma-data:
```

After that, create a **.env** file like the below script (The value must be the same as the value you ran the query in MariaDB):

```
MARIADB_DATABASE=uptime_kuma
MARIADB_USER=kuma-user
MARIADB_PASSWORD=kumapass123
MARIADB_ROOT_PASSWORD=qwerty
```

Run the below command to turn on the docker compose:

```
docker compose up -d
```

To check if the containers are running or not, use the command below:

```
docker ps
```

After you type the commands, your screen will show up similar to the one below:



Check the running container

3. Configure webserver

If you use Apache, create a file at **/etc/apache2/sites-available/kuma.conf** and copy the script below to the file:

```
<VirtualHost *:80>
ServerName yourdomain.com
DocumentRoot /var/www/html/

ProxyPass / http://localhost:3001/
RewriteEngine on
RewriteCond %{HTTP:Upgrade} websocket [NC]
RewriteCond %{HTTP:Connection} upgrade [NC]
RewriteRule ^/?(.*) "ws://localhost:3001/$1" [P,L]

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

</VirtualHost>
```

then run the command below:

```
sudo a2enmod rewrite
sudo a2enmod proxy
sudo a2enmod proxy_http
sudo a2ensite kuma.conf
```

Check if there is an error in Apache and if there is no error, reload Apache using the command below:

```
apachectl -t
sudo systemctl reload apache2
```

INFO

If your server is running an nginx webserver, then in the file **/etc/nginx/conf.d/uptime-kuma.conf** insert the script below:

```
server {
    listen 80;
    server_name uptime-kuma.yourdomainname.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;
        proxy_set_header   X-Real-IP $remote_addr;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header   X-Forwarded-Proto $scheme;

        # Added WebSocket support
        proxy_set_header   Sec-WebSocket-Key $http_sec_websocket_key;
        proxy_set_header   Sec-WebSocket-Version $http_sec_websocket_version;
        proxy_set_header   Sec-WebSocket-Extensions
$http_sec_websocket_extensions;

        # Improve performance of this reverse proxy
        proxy_buffering    off;
    }

    # Redirect HTTP to HTTPS if needed for encryption
    # Uncomment the following lines if you have SSL enabled
    # return 301 https://$host$request_uri;
}
```

Use the command below to check if there is an error in the nginx configuration and then reload nginx:

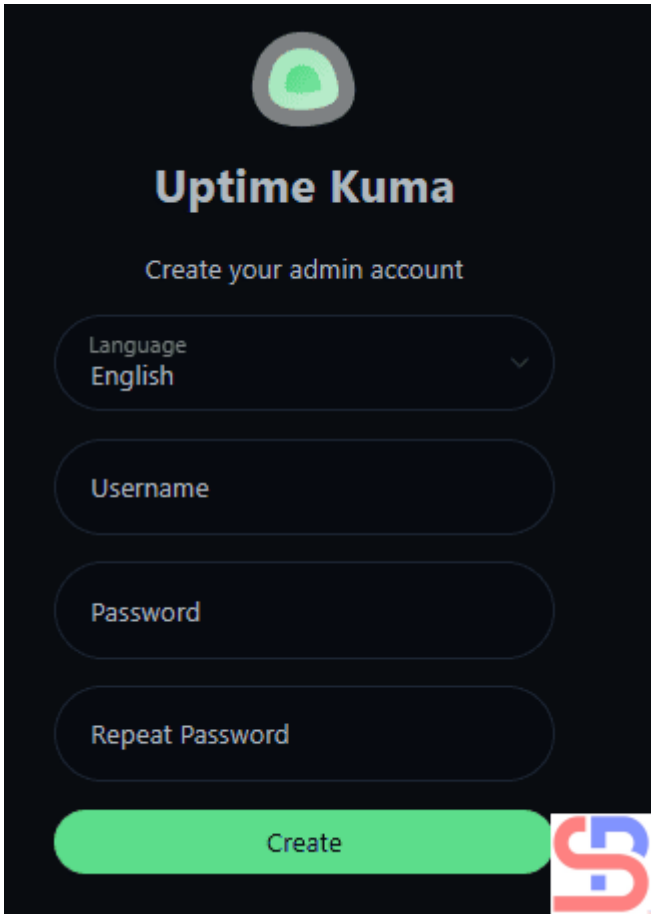
```
sudo nginx -t
sudo systemctl reload nginx
```

4. Access uptime kuma

Open your browser, and type:

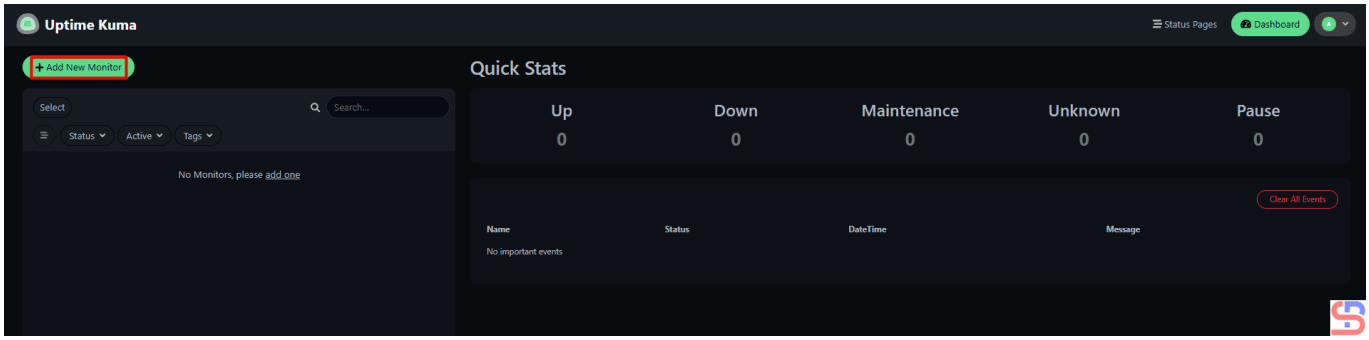
```
http://ip_server:3001
```

then there will be a display like below:



Fill in the columns for the admin account

Enter in the columns above the value you want and press the **Create** button then a display will appear similar to the image provided below:



Display of uptime kuma

If you want to make sure uptime kuma use MariaDB database, run the command below:

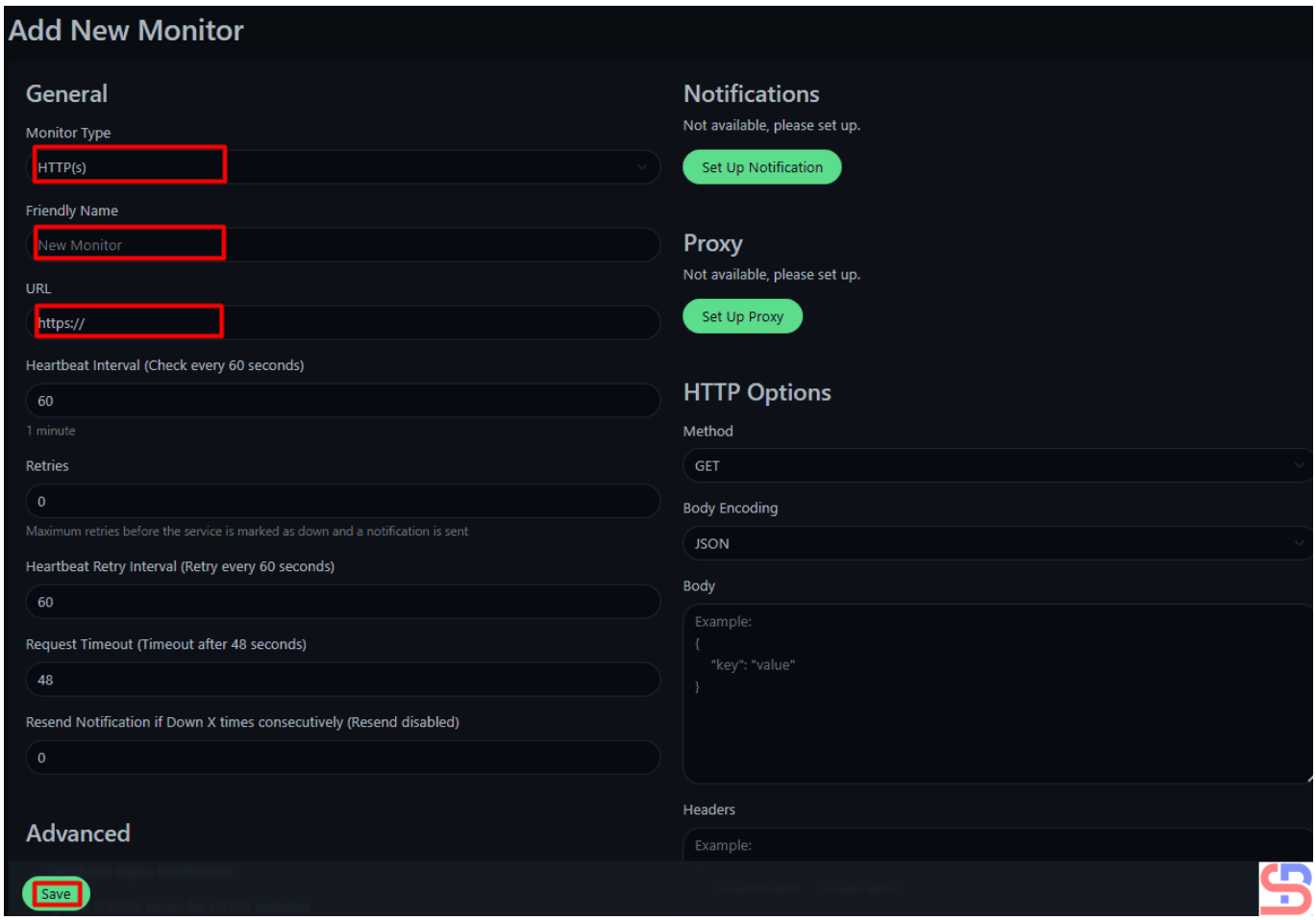
```
docker logs uptime-kuma | grep DB
```

Your screen will appear similar to the picture shown below:



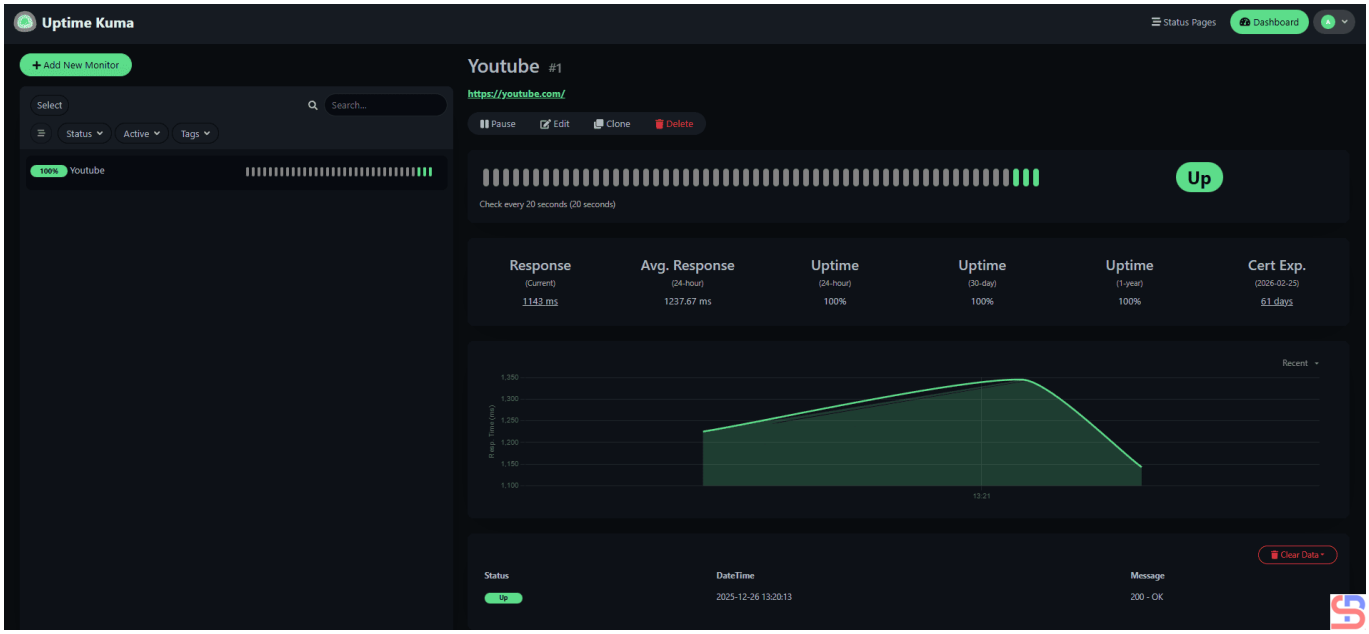
Check the running database

If you want to monitor the website, click the **Add New Monitor** button at the top left, an image similar to the one shown will appear:



Create a new host or a website to monitor in uptime kuma

Fill in the required fields (at least fill in the **Monitor Type**, **Friendly Name**, and **URL** columns) and press the **Save** button, then the host you have filled in will look like in the image below:



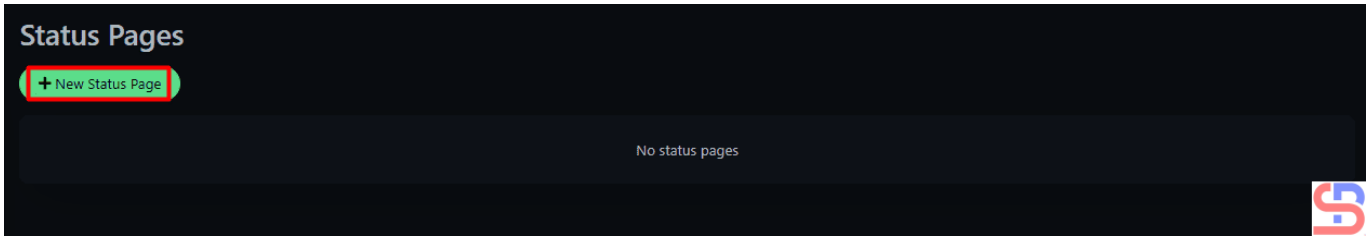
Monitor the host or the website

If you just want to display the status without displaying many attributes then you can click the **Status Pages** button at the top right of the site like the image below:



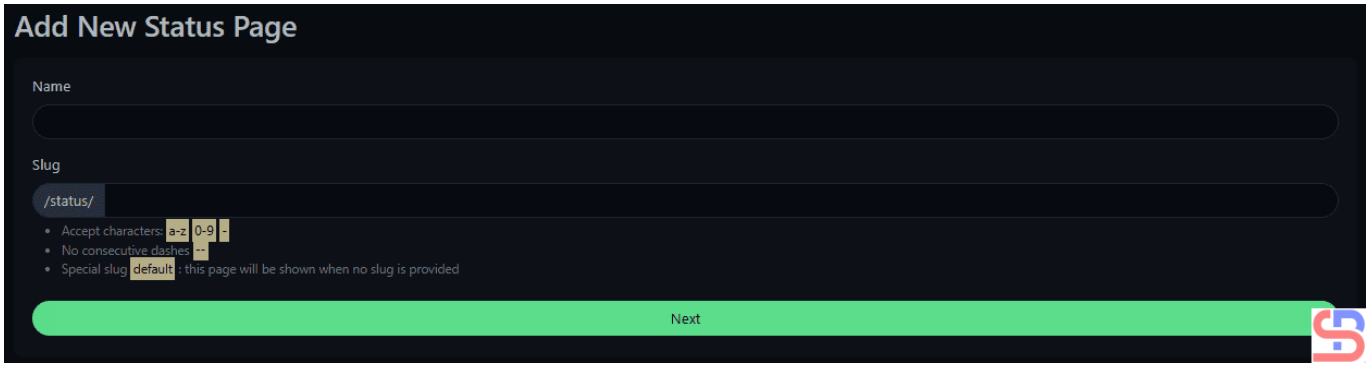
Click the Status Page button

After you press the Status Page button, the following image will appear:



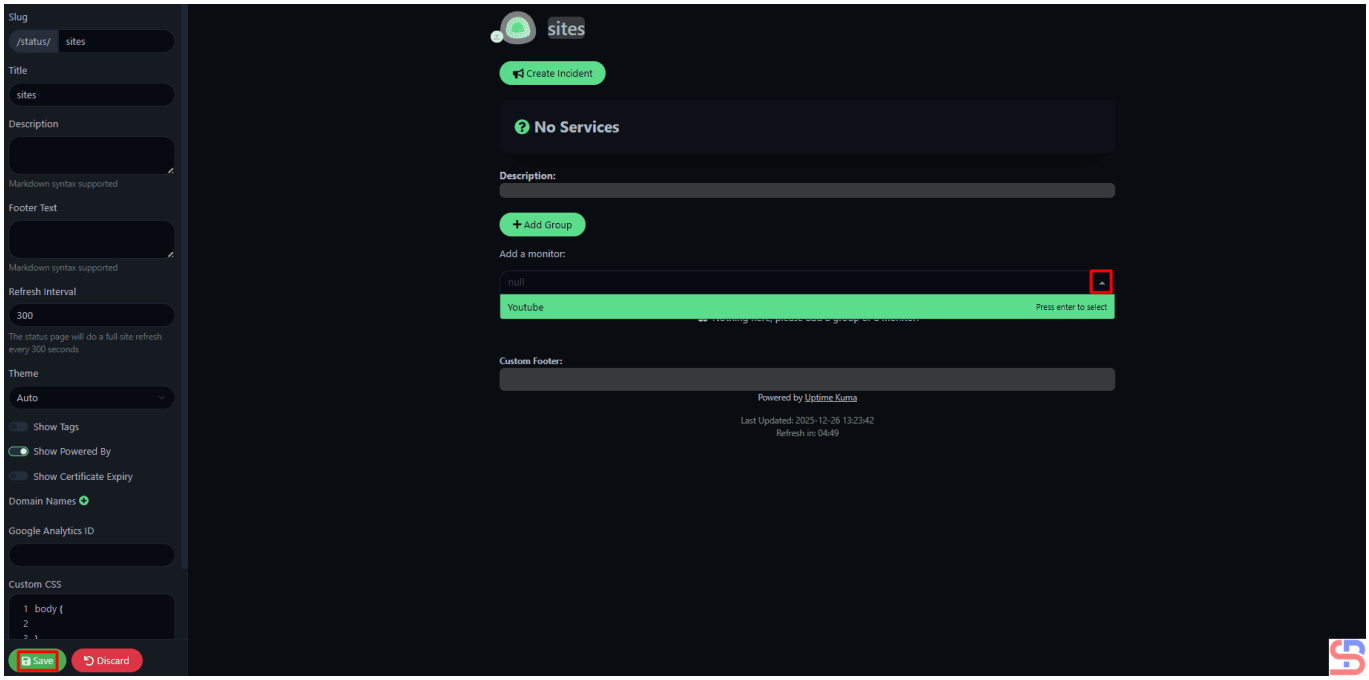
Create the Status Page page

Click the **New Status Page** button, and an image will appear similar to the one shown below:



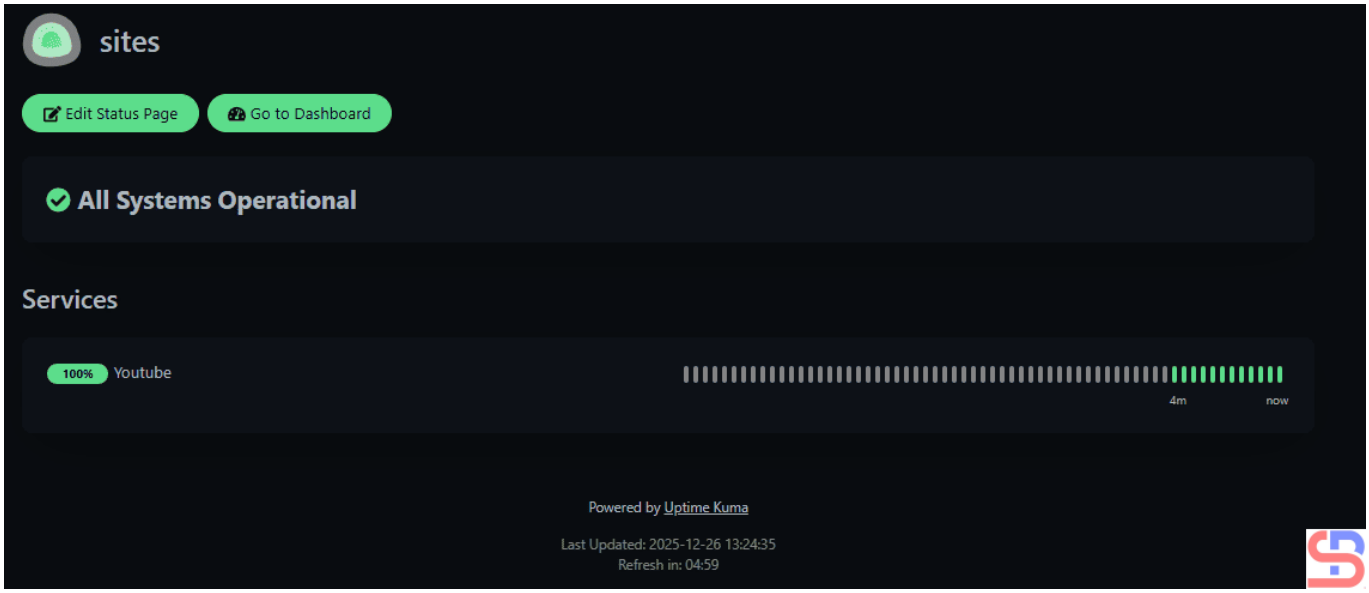
Create the Status Page page

Enter the name and slug you want (I wrote the sites for the name and slug), then press the Next button, then there will be a display as below:



Insert the host or the monitor in the Status Page

Enter the host you want to display on the Status Page, after that click the Save button, then there will be a display as below:



Display of Status Page

You can see that the hosts to be monitored look simpler and you can give the URL to other parties to also monitor these hosts.

Note

Go to this page if you want to backup the MariaDB database and for how to restore the database, you can go to this page.

References

sysadminpedia.com
magnus919.com
uptimekuma.org

How to Backup And Restore Uptime Kuma Database in Docker?

written by sysadmin | 17 January 2026

The previous article explained how to install Uptime Kuma using Docker on Linux. This article will explain how to back up and restore the Kuma uptime database in Docker.

Problem

How to back up and restore the Uptime Kuma database in Docker?

Solution

Below are the steps to back up and restore the Uptime Kuma database in Docker:

A. Database SQLite

Here is the method to back up and restore a SQLite database in Docker:

1. Backup database

If you want to back up the Uptime Kuma database in Docker, you can run the command below to get the Kuma database on your host:

```
docker run --rm
  -v uptime-kuma:/data
  -v $(pwd):/backup
  alpine tar czf /backup/kuma-backup.tar.gz /data
```

After that, look in your current folder; the database should appear like the image below:

```
sysadmin@docker:~$ docker run --rm \
  -v uptime-kuma:/data \
  -v $(pwd):/backup \
  alpine tar czf /backup/kuma-backup.tar.gz /data
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
1074353eec0d: Pull complete
5c1f58ba4e0d: Download complete
644afed44dca: Download complete
Digest: sha256:865b95f46d98cf867a156fe4a135ad3fe50d2056aa3f25ed31662dff6da4eb62
Status: Downloaded newer image for alpine:latest
tar: removing leading '/' from member names
sysadmin@docker:~$ ls
crud  get-docker.sh  kuma-backup.tar.gz  main.zip
```

Back up the uptime Kuma database

However, if you want to back up automatically, then follow the steps below:

a. Create a backup folder on the host

Run the commands below to create a backup folder in the host:

```
sudo mkdir -p /opt/kuma-backup
sudo chown root:root /opt/kuma-backup
sudo chmod 700 /opt/kuma-backup
```

b. Run the backup script

Create a file, for example, backup-kuma.sh, in the root folder containing the bash script below:

```
#!/bin/bash
set -e

# CONFIG
CONTAINER_NAME="uptime-kuma"
VOLUME_NAME="uptime-kuma"
BACKUP_DIR="/opt/kuma-backup"
RETENTION_DAYS=14
DATE=$(date +"%Y%m%d-%H%M%S")
BACKUP_FILE="$BACKUP_DIR/kuma-backup-$DATE.tar.gz"

# Backup
docker run --rm \
  -v $VOLUME_NAME:/data:ro \
```

```
  -v $BACKUP_DIR:/backup \
  alpine \
  tar czf /backup/$(basename $BACKUP_FILE) /data

# Cleanup old backups
find $BACKUP_DIR -type f -name "kuma-backup-*.tar.gz" -mtime +$RETENTION_DAYS
-delete
```

Save the file and make it executable using the command:

```
sudo chmod +x /root/backup-kuma.sh
```

If you run the script, the backup database file should be in the **/opt/backup** folder.

c. Set up a Cron Job

If you want to run the script every 2:30 AM, then on the crontab, write the script as shown in the image below

```
0 2 * * *       /root/backup-kuma.sh >> /root/kuma-backup.log 2>&1
```

2. Restore database

Before you restore the database, make sure the container has been running first. If you want to restore the Uptime Kuma database that you have previously backed up, then you can run the command below on the host:

```
docker run --rm \
  -v uptime-kuma:/data \
  -v /opt/kuma-backup:/backup \
  alpine \
  tar xzf /backup/kuma-backup-YYYYMMDD-HHMMSS.tar.gz -C /
```

If the restore process is complete, the hosts that were monitored in the previous container should be monitored again by the new container.

B. Database MariaDB

Here is the method to back up and restore a MariaDB database

in Docker:

1. Backup database

If you want to back up the MariaDB database in Docker, you can run the command below to get the database on your host:

```
docker exec mariadb \
  mysqldump -u root -p kuma > kuma.sql
```

Change the kuma with your database name. Or use the command below if you want to compress the result:

```
docker exec mariadb \
  mysqldump -u root -p kuma | gzip > kuma.sql.gz
```

You can use the script below to back up the database:

```
#!/bin/bash

# ========================
# CONFIGURATION
# ========================
CONTAINER_NAME="mariadb"
DB_USER="backup_user"
DB_PASSWORD="PASSWORD_DB"
DB_NAME="appdb"

BACKUP_DIR="/opt/backup/mariadb"
RETENTION_DAYS=7
DATE=$(date +"%Y-%m-%d_%H-%M-%S")
BACKUP_FILE="$BACKUP_DIR/${DB_NAME}_${DATE}.sql.gz"
LOG_FILE="$BACKUP_DIR/backup.log"

# ========================
# PREPARATION
# ========================
mkdir -p "$BACKUP_DIR"

echo "[$(date)] Backup started" >> "$LOG_FILE"

# ========================
# BACKUP DATABASE
# ========================
docker exec "$CONTAINER_NAME" \
  mysqldump -u"$DB_USER" -p"$DB_PASSWORD" "$DB_NAME" \
  --single-transaction \
```

```
  --quick \
  --routines \
  --triggers \
  --events \
  | gzip > "$BACKUP_FILE"

# ========================
# VALIDATION
# ========================
if [ $? -eq 0 ]; then
  echo "[$(date)] Backup SUCCESS: $BACKUP_FILE" >> "$LOG_FILE"
else
  echo "[$(date)] Backup FAILED" >> "$LOG_FILE"
  exit 1
fi

# ========================
# BACKUP ROTATION (DELETE OLD FILE)
# ========================
find "$BACKUP_DIR" -type f -name "*.sql.gz" -mtime +$RETENTION_DAYS -delete

echo "[$(date)] Old backups cleaned (>${RETENTION_DAYS} days)" >> "$LOG_FILE"
echo "[$(date)] Backup finished" >> "$LOG_FILE"
```

And you can insert the script above in the crontab.

2. Restore database

Before you restore the database, make sure the container has been running first. If you want to restore the database that you have previously backed up, then you can run the command below on the host:

```
docker exec -i mariadb \
  mysql -u root -p -e "CREATE DATABASE kuma;"

docker exec -i mariadb \
  mysql -u root -p kuma < kuma.sql
```

Note

If you have a failure to back up the database, you can see the logs in the **/root/backup-kuma.log** file.

References

fishparts.net
homelab.anita-fred.net

How to Install Uptime Kuma And MariaDB in Docker?

written by sysadmin | 17 January 2026

The previous article explained how to install the Uptime Kuma application using Docker. However, by default, Uptime Kuma uses a SQLite database, and you want to change the database to MariaDB for some reasons.

Problem

How to install Uptime Kuma and MariaDB in Docker?

Solution

Although SQLite serves as a superb embedded database option for numerous scenarios, certain inherent limitations render it inappropriate for particular applications. If your application has a large amount of traffic and uses a lot of write modes simultaneously, and the data growth is very fast, your application is not suitable for using a SQLite database. Likewise, with the Uptime Kuma application. If you monitor many hosts using low intervals, it will cause very fast data growth, so you have to think about another database solution besides the SQLite database.

1. Create a Docker Compose file

Create a compose folder in the **/opt** folder using the command below:

```
sudo mkdir -p /opt/compose/uptime-kuma/
cd /opt/compose/uptime-kuma/
```

After that, create **docker-compose.yaml** file and copy the script below:

```
services:
  mariadb:
    image: mariadb:11.4
    container_name: mariadb
    restart: unless-stopped
    environment:
      MARIADB_ROOT_PASSWORD: ${MARIADB_ROOT_PASSWORD}
      MARIADB_DATABASE: ${MARIADB_DATABASE}
      MARIADB_USER: ${MARIADB_USER}
      MARIADB_PASSWORD: ${MARIADB_PASSWORD}
    volumes:
      - mariadb-data:/var/lib/mysql
    networks:
      - kuma-net
    healthcheck:
      test: ["CMD", "healthcheck.sh", "--connect", "--innodb_initialized"]
      interval: 10s
      timeout: 5s
      retries: 5

  uptime-kuma:
    image: louislam/uptime-kuma:2
    container_name: uptime-kuma
    restart: unless-stopped
    depends_on:
      mariadb:
        condition: service_healthy
    ports:
      - "3001:3001"
    volumes:
      - kuma-data:/app/data
    networks:
      - kuma-net

volumes:
  mariadb-data:
  kuma-data:

networks:
  kuma-net:
```

After that, create a **.env** file like the below script (Adjust the value of this file to your liking):

```
MARIADB_DATABASE=kuma
MARIADB_USER=kuma-user
MARIADB_PASSWORD=123456
MARIADB_ROOT_PASSWORD=qwerty
```

Run the below command to turn on Docker Compose:

```
docker compose up -d
```

To check if the containers are running or not, use the command below:

```
docker ps
```

After you type the commands, your screen will show up similar to the one below:



Run the Docker Compose

2. Configure the web server

If you use Apache, create a file at **/etc/apache2/sites-available/kuma.conf** and copy the script below to the file:

```
<VirtualHost *:80>
ServerName yourdomain.com
DocumentRoot /var/www/html/

ProxyPass / http://localhost:3001/
RewriteEngine on
RewriteCond %{HTTP:Upgrade} websocket [NC]
RewriteCond %{HTTP:Connection} upgrade [NC]
RewriteRule ^/?(.*) "ws://localhost:3001/$1" [P,L]
```

```
ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

</VirtualHost>
```

Then run the command below:

```
sudo a2enmod rewrite
sudo a2enmod proxy
sudo a2enmod proxy_http
sudo a2ensite kuma.conf
```

Check if there is an error in Apache, and if there is no error, reload Apache using the command below:

```
apachectl -t
sudo systemctl reload apache2
```

INFO

If your server is running an nginx webserver, then in the file **/etc/nginx/conf.d/uptime-kuma.conf** insert the script below:

```
server {
    listen 80;
    server_name uptime-kuma.yourdomainname.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;
        proxy_set_header   X-Real-IP $remote_addr;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header   X-Forwarded-Proto $scheme;

        # Added WebSocket support
        proxy_set_header   Sec-WebSocket-Key $http_sec_websocket_key;
        proxy_set_header   Sec-WebSocket-Version $http_sec_websocket_version;
        proxy_set_header   Sec-WebSocket-Extensions
$http_sec_websocket_extensions;

        # Improve performance of this reverse proxy
        proxy_buffering    off;
    }
```

```
    # Redirect HTTP to HTTPS if needed for encryption
    # Uncomment the following lines if you have SSL enabled
    # return 301 https://$host$request_uri;
}
```

Use the command below to check if there is an error in the nginx configuration and then reload nginx:

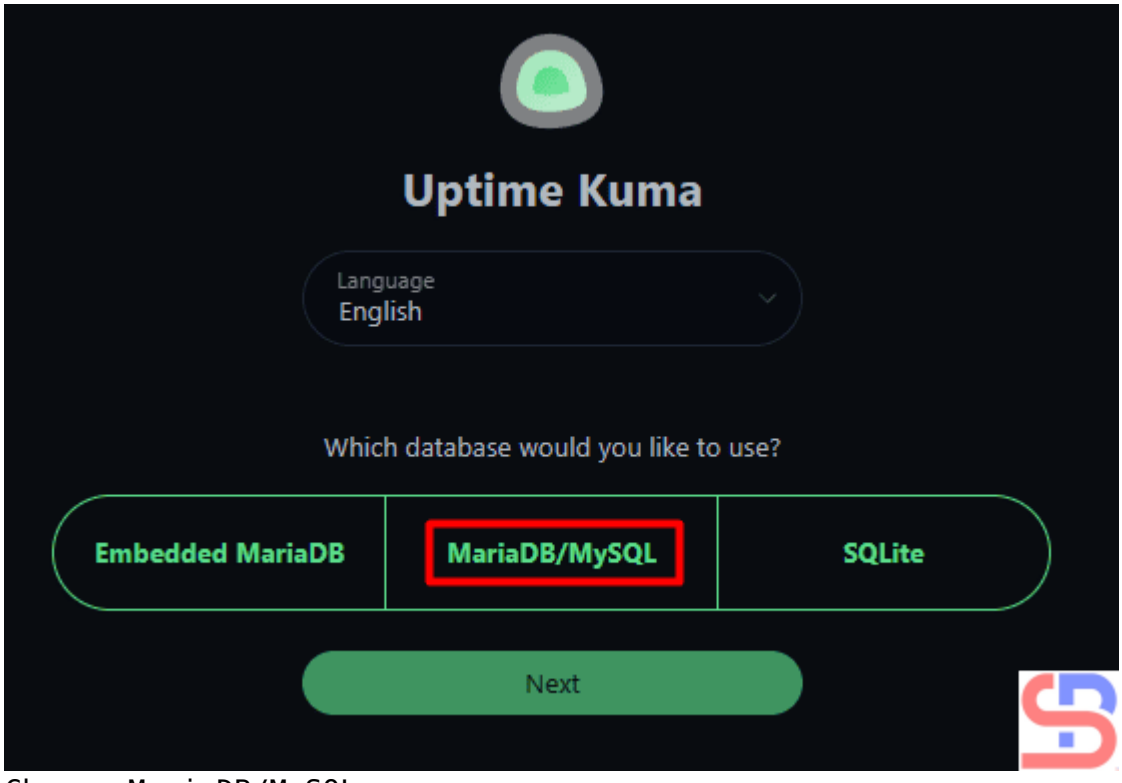
```
sudo nginx -t
sudo systemctl reload nginx
```

3. Access uptime kuma

Open your browser, and type:

```
http://ip_server:3001
```

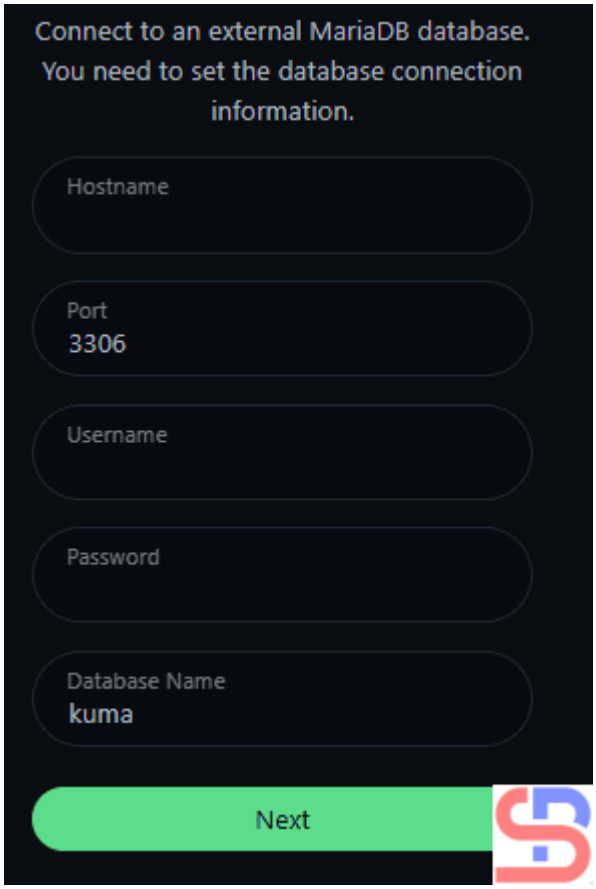
Then there will be a display like below:



Choose MariaDB/MySQL

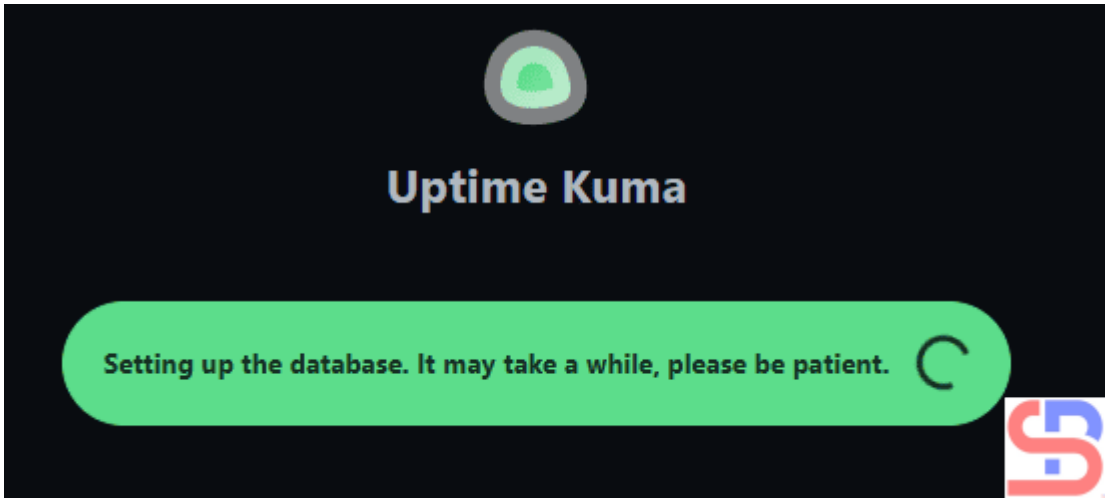
Click **MariaDB/MySQL**, and your screen will appear similar to

the picture below:



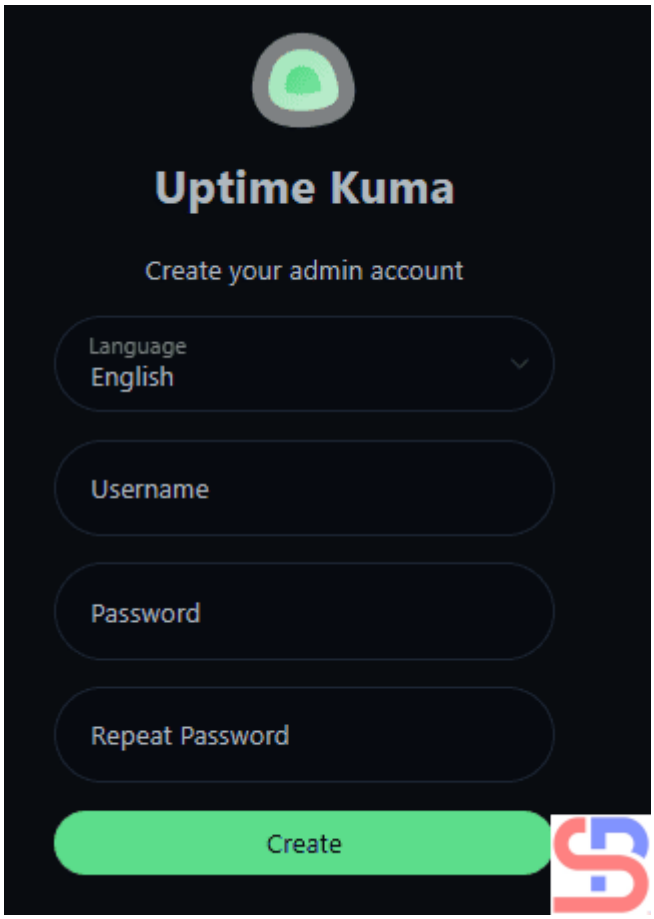
Fill in the columns for the database

Enter in the columns above the values □□that correspond to the **.env** file. Click the **Next** button, and your screen will show up similar to the one below:



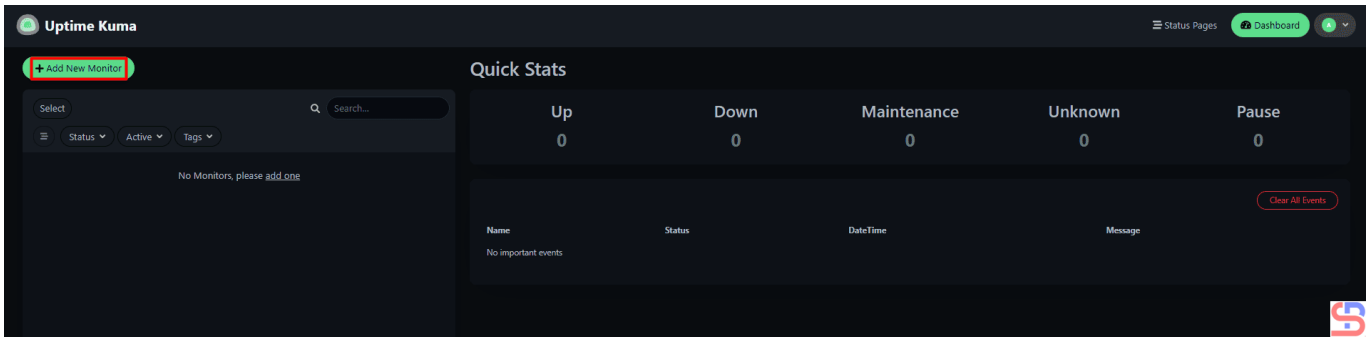
Setting up the database

You have to wait until finishes, and after that, your screen will appear similar to the image shown below:



Fill in the columns for the admin account

Enter in the columns above the value you want and press the **Create** button, then a display will appear similar to the image provided below:

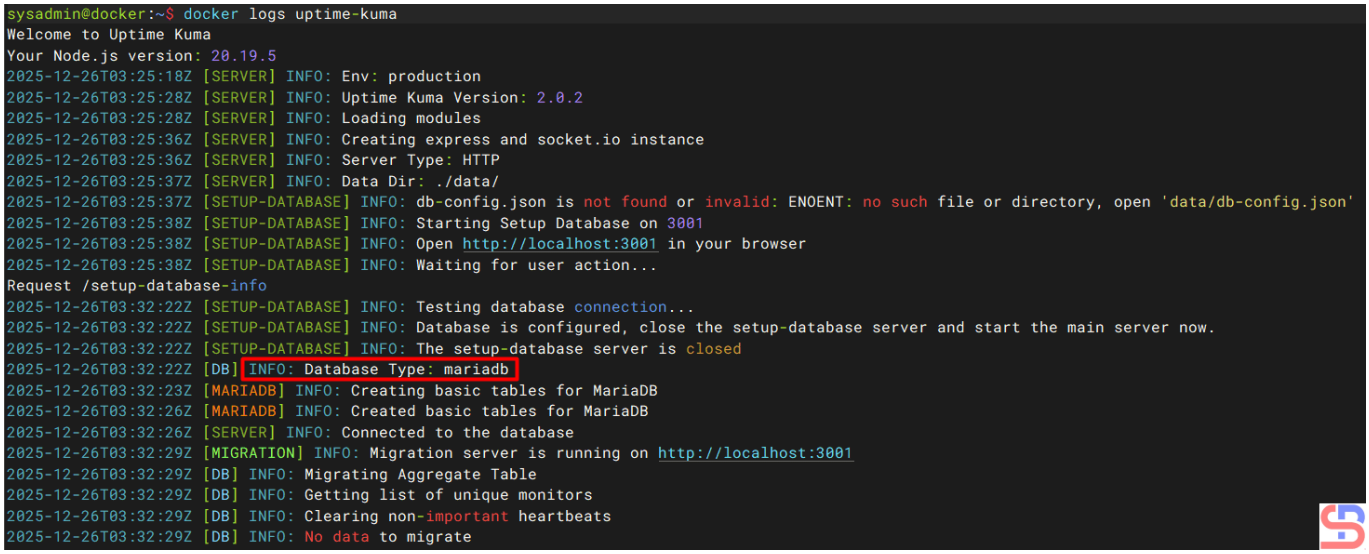


Display of uptime kuma

If you want to make sure Uptime Kuma uses a MariaDB database, run the command below:

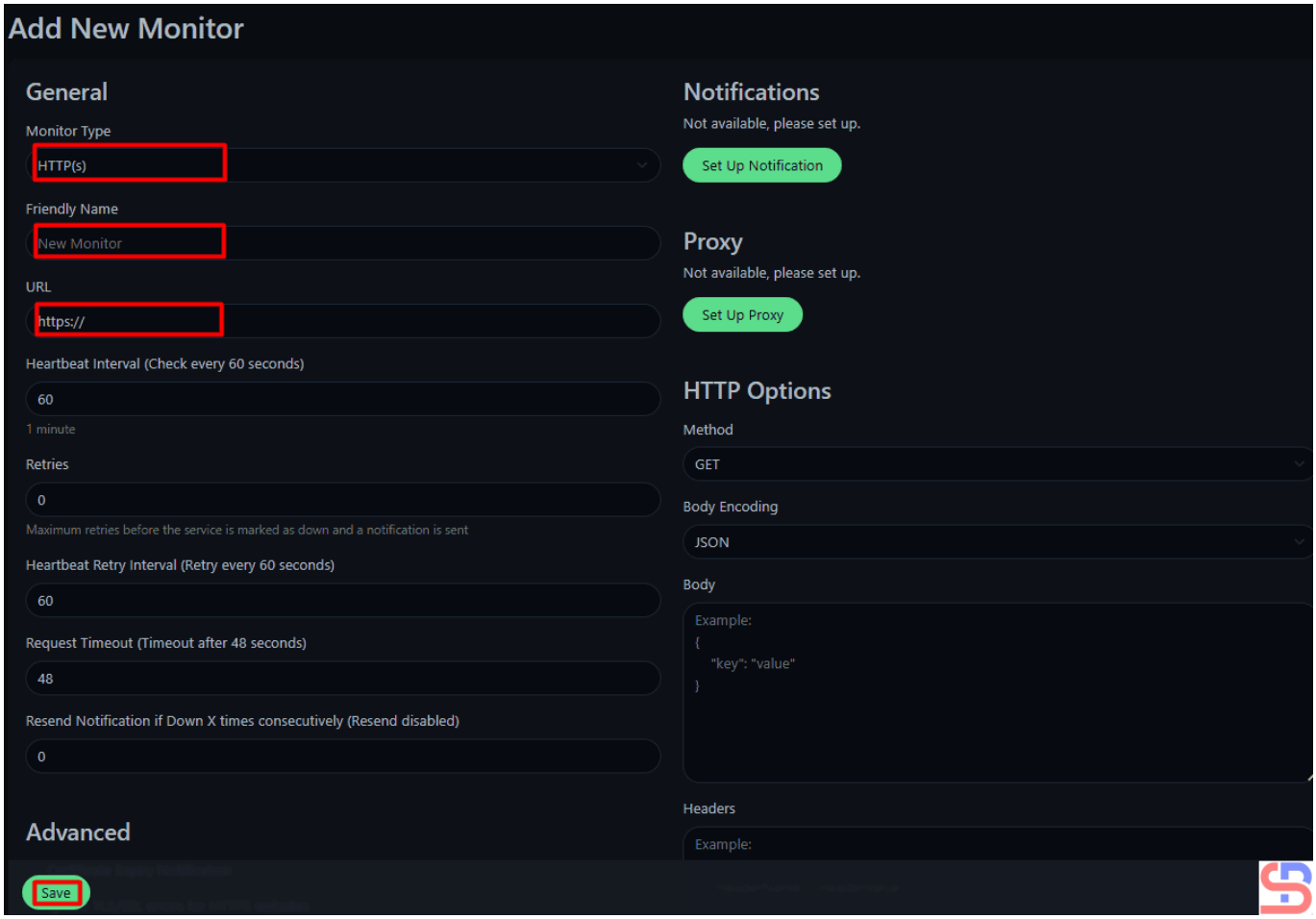
```
docker logs uptime-kuma | grep DB
```

Your screen will appear similar to the picture shown below:



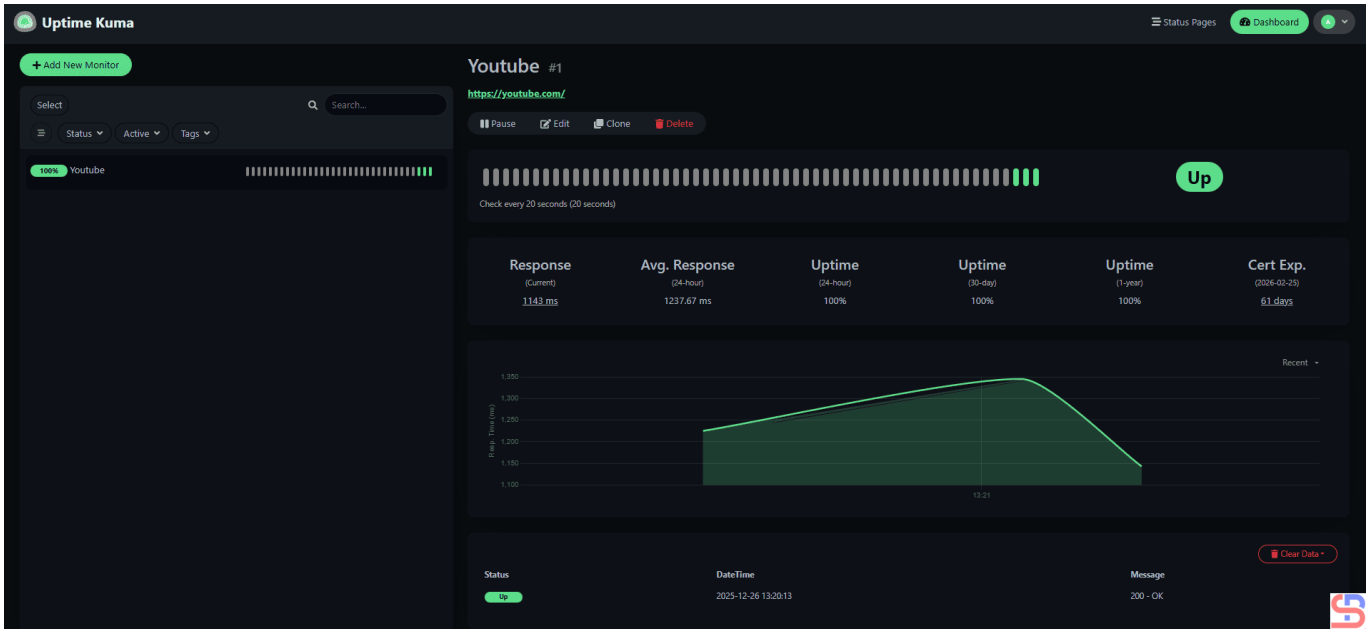
Check the running database

If you want to monitor the website, click the **Add New Monitor** button at the top left of the site, and an image similar to the one shown will appear:



Create a new host or a website to monitor in Uptime Kuma

Fill in the required fields (at least fill in the **Monitor Type**, **Friendly Name**, and **URL** columns) and press the **Save** button, then the host you have filled in will look like in the image below:



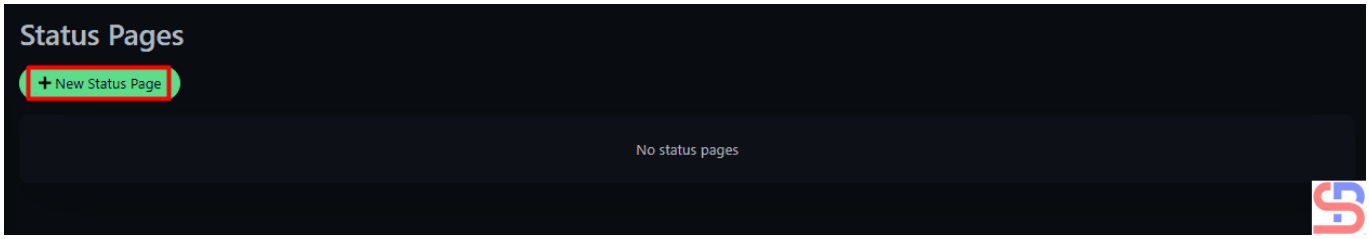
Monitor the host or the website

If you just want to display the status without displaying many attributes, then you can click the **Status Pages** button at the top right of the site, like the image below:



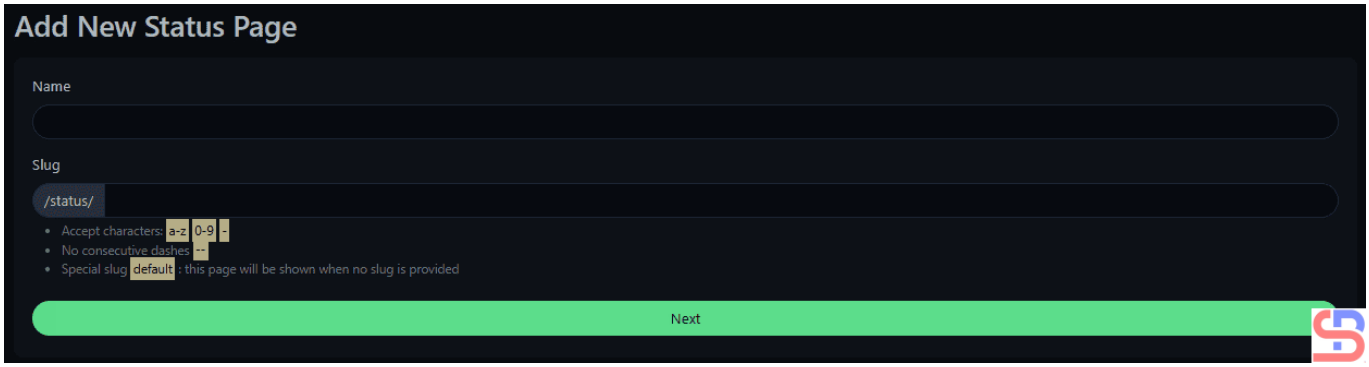
Click the Status Page button

After you press the Status Page button, the following image will appear:



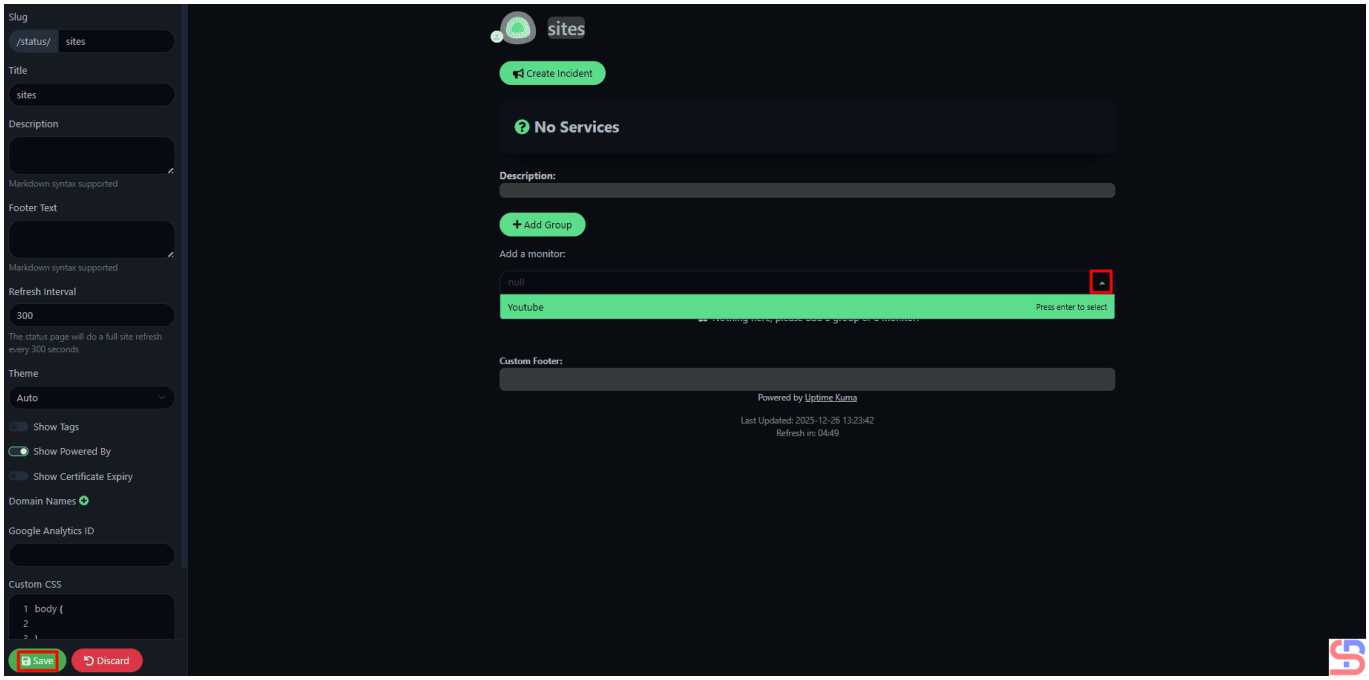
Create the Status Page page

Click the **New Status Page** button, and an image will appear similar to the one shown below:



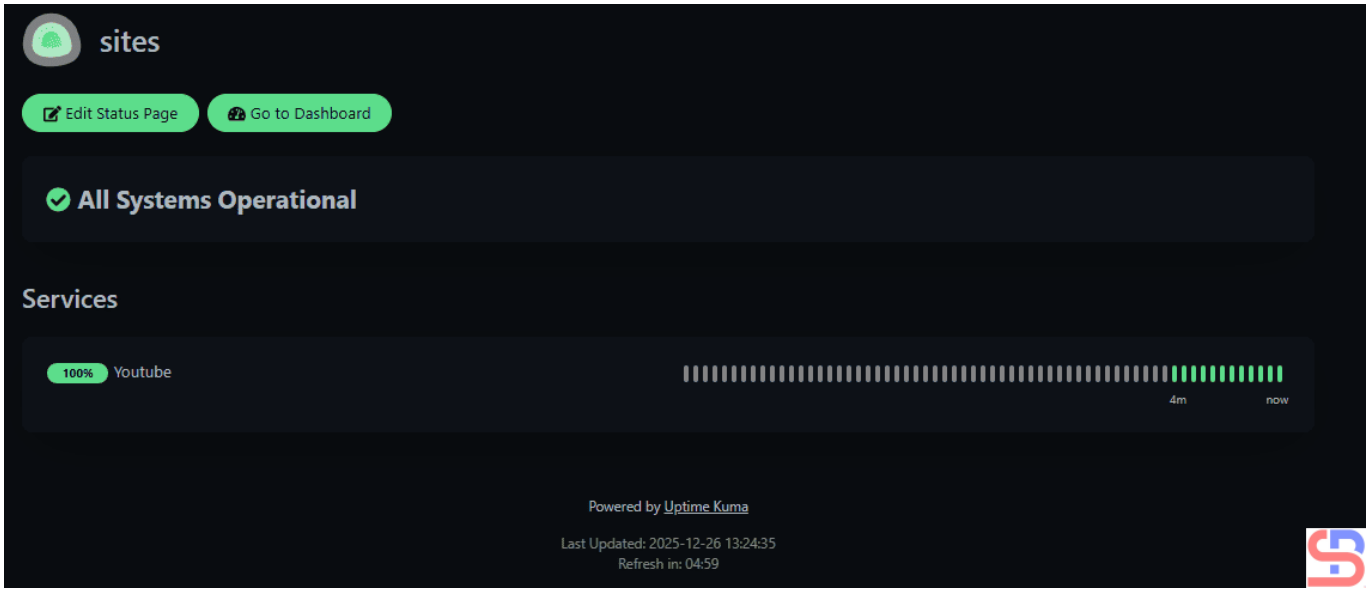
Create the Status Page page

Enter the name and slug you want (I wrote the sites for the name and slug), then press the Next button, and then there will be a display as below:



Insert the host or the monitor in the Status Page

Enter the host you want to display on the Status Page, after that, click the Save button, then there will be a display as below:



Display of Status Page

You can see that the hosts to be monitored look simpler, and you can give the URL to other parties to also monitor these hosts.

Note

If you want to back up the MariaDB database running on Docker and learn how to restore the database, you can go to this page.

References

quora.com
magnus919.com
uptimekuma.org

How to Install Uptime Kuma on Linux?

written by sysadmin | 17 January 2026

Uptime Kuma is a self-hosted monitoring solution created to measure the uptime and performance of websites and services. It offers live status updates, flexible alerting choices, and comprehensive metrics to help guarantee that your websites and services stay functional.

Problem

How to install Uptime Kuma on Linux?

Solution

There are 4 methods to install uptime kuma:

1. Using docker.
2. Using docker compose with database in docker.
3. Using docker compose with database in the host.
4. Using package.

This article will explain how to install Kuma using Docker.

1. Install uptime kuma

Make sure you installed Docker in your server and you can see how to install Docker on this page. After that, run the command below to install uptime kuma using docker:

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

Then check whether the uptime kuma container is running or not using the command:

```
docker ps | grep Kuma
```



Check the uptime kuma container in Docker

2. Configure webserver

If you use Apache, create a file at **/etc/apache2/sites-available/kuma.conf** and copy the script below to the file:

```
<VirtualHost *:80>
ServerName yourdomain.com
DocumentRoot /var/www/html/

ProxyPass / http://localhost:3001/
RewriteEngine on
RewriteCond %{HTTP:Upgrade} websocket [NC]
RewriteCond %{HTTP:Connection} upgrade [NC]
RewriteRule ^/?(.*) "ws://localhost:3001/$1" [P,L]

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

</VirtualHost>
```

then run the command below:

```
sudo a2enmod rewrite
sudo a2enmod proxy
sudo a2enmod proxy_http
sudo a2ensite kuma.conf
```

Check if there is an error in apache and if there is no error, reload apache using the command below:

```
apachectl -t
sudo systemctl reload apache2
```

INFO

If your server is running an nginx webserver, then in the file **/etc/nginx/conf.d/uptime-kuma.conf** insert the script below:

```
server {
    listen 80;
    server_name uptime-kuma.yourdomainname.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;
        proxy_set_header   X-Real-IP $remote_addr;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header   X-Forwarded-Proto $scheme;

        # Added WebSocket support
        proxy_set_header   Sec-WebSocket-Key $http_sec_websocket_key;
        proxy_set_header   Sec-WebSocket-Version $http_sec_websocket_version;
        proxy_set_header   Sec-WebSocket-Extensions
$http_sec_websocket_extensions;

        # Improve performance of this reverse proxy
        proxy_buffering    off;
    }

    # Redirect HTTP to HTTPS if needed for encryption
    # Uncomment the following lines if you have SSL enabled
    # return 301 https://$host$request_uri;
}
```

Use the command below to check if there is an error in the nginx configuration and then reload nginx:

```
nginx -t
sudo systemctl reload nginx
```

3. Configure database

If you install uptime kuma using docker, you **don't need to install the database** because in the docker there is already a SQLite database where you can view it by using the command below:

```
docker exec -it uptime-kuma ls /app/data
```

```
root@docker:~# docker exec -it uptime-kuma ls /app/data
docker-tls  kuma.db  kuma.db-shm  kuma.db-wal  screenshots  upload
root@docker:~#
```

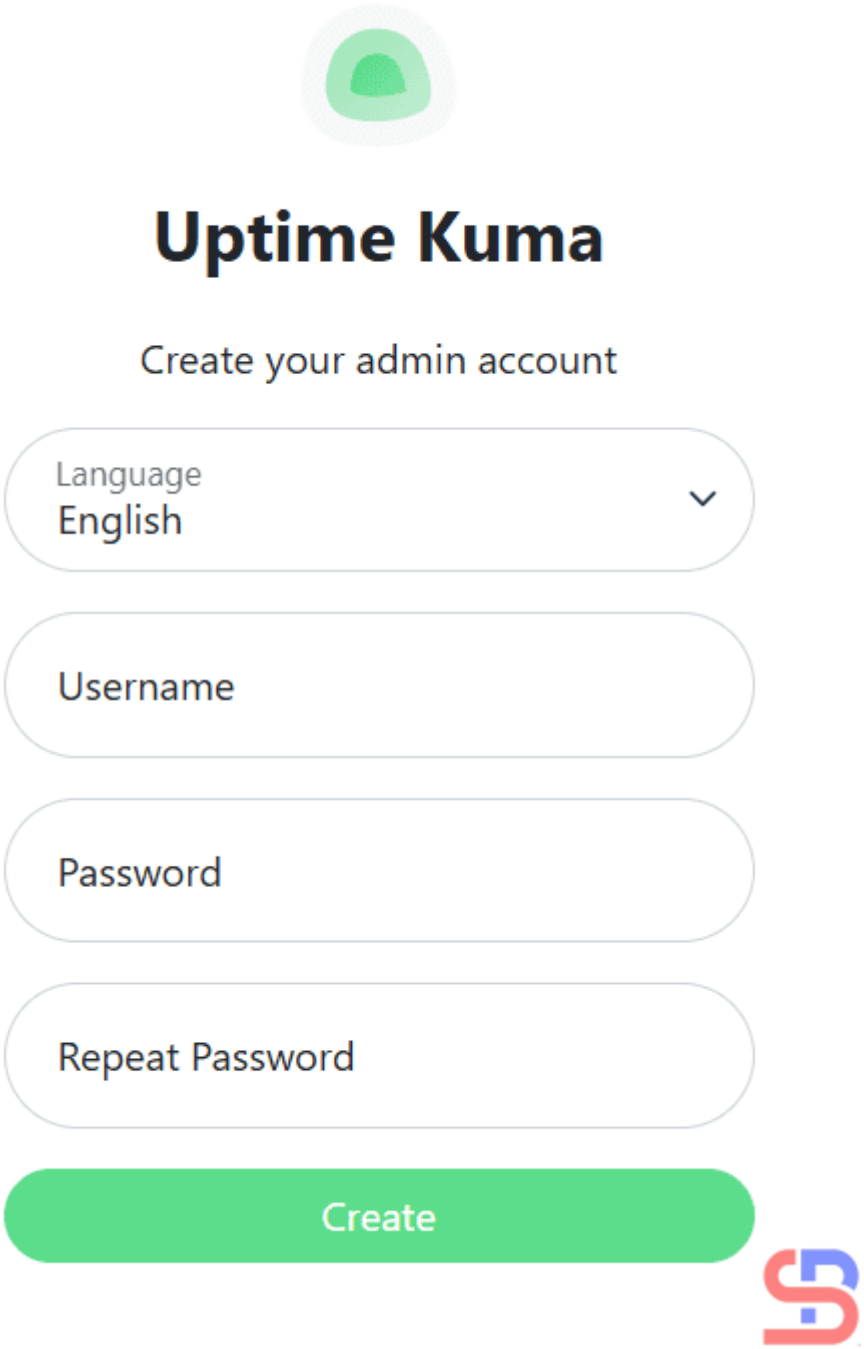
Check the database in the uptime kuma container

4. Access uptime kuma

Open your browser, and type:

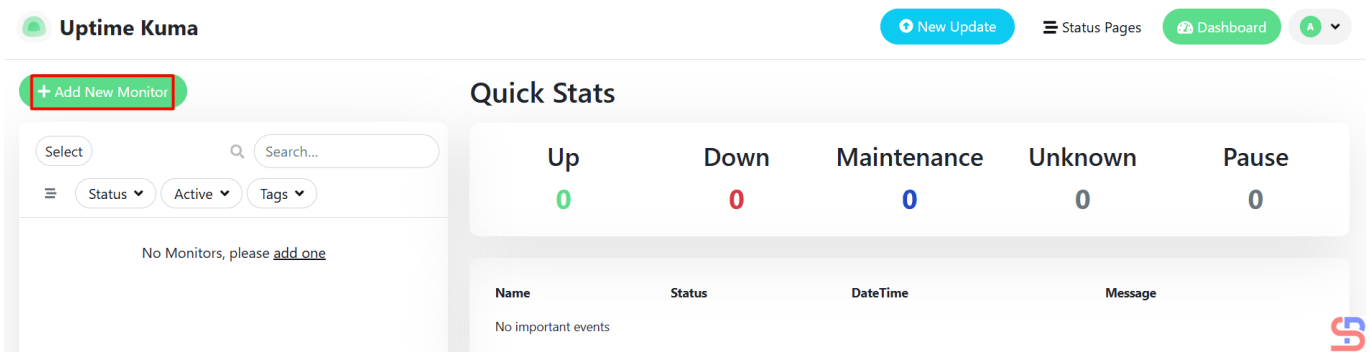
```
http://ip_server:3001
```

then there will be a display like below:



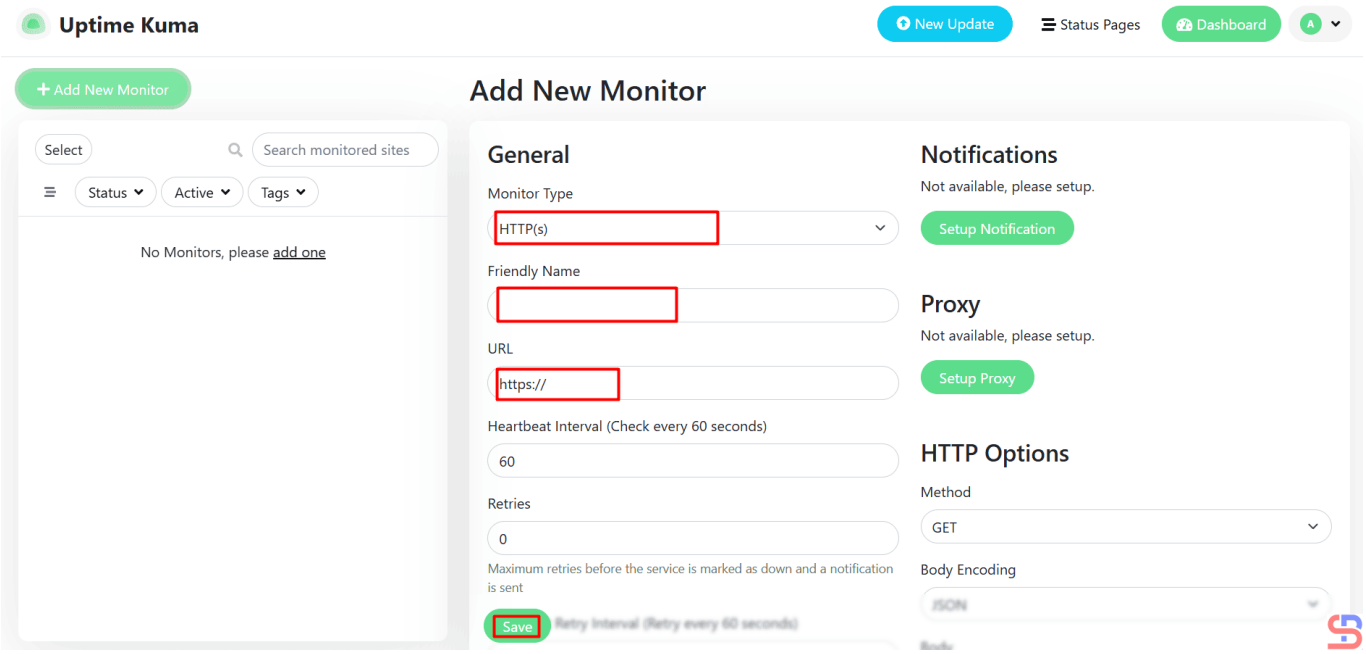
Create username and password for Uptime Kuma

Enter the username and password you want then press the **Create** button, there will be a display as below:



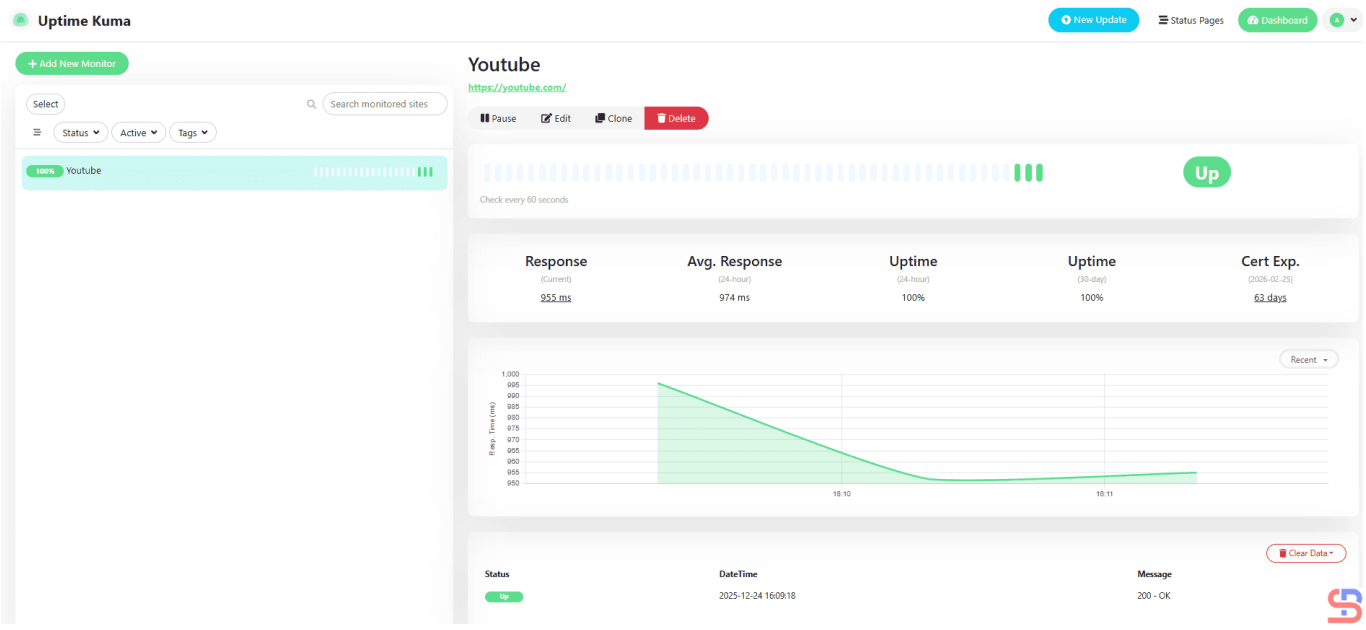
Display of uptime kema application

If you want to monitor a host or a website, click the **Add New Monitor** button like in the below image:



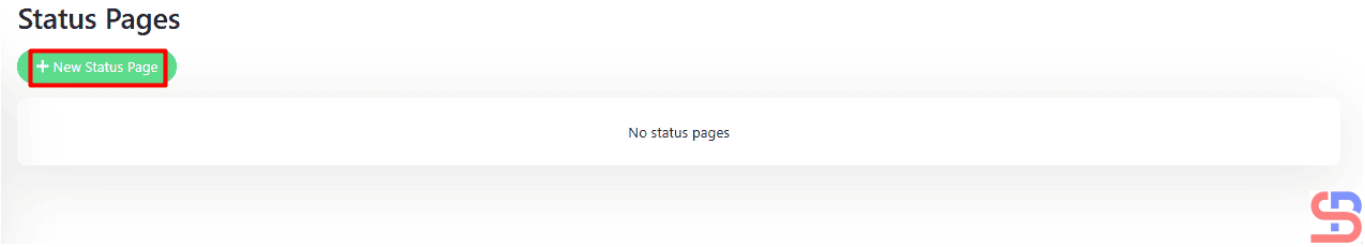
Create a new host or a website to monitor in uptime kuma

Fill in the required fields (at least fill in the **Monitor Type**, **Friendly Name**, and **URL** columns) and press the **Save** button, then the host you have filled in will look like in the image below:



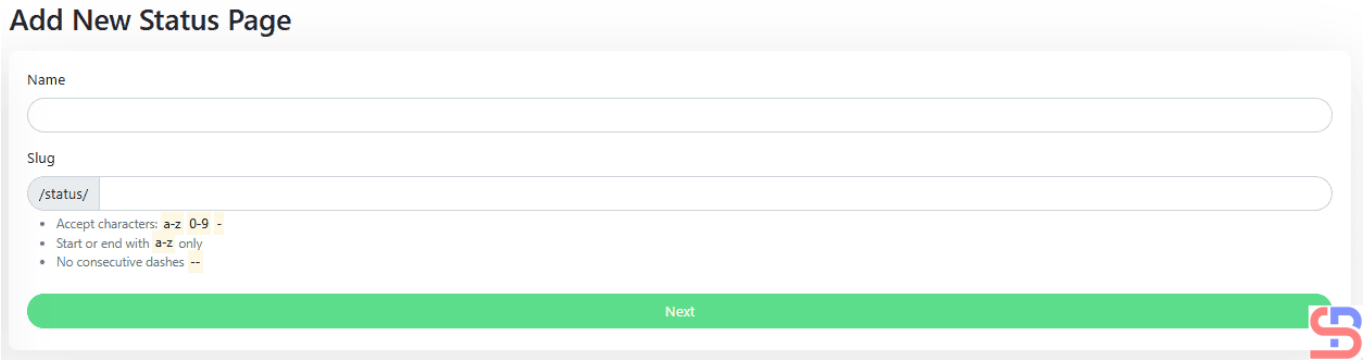
Monitor the host or the website

If you just want to display the status without displaying many attributes then you can click the **Status Pages** button at the top right of the layer then there will be a display like below:



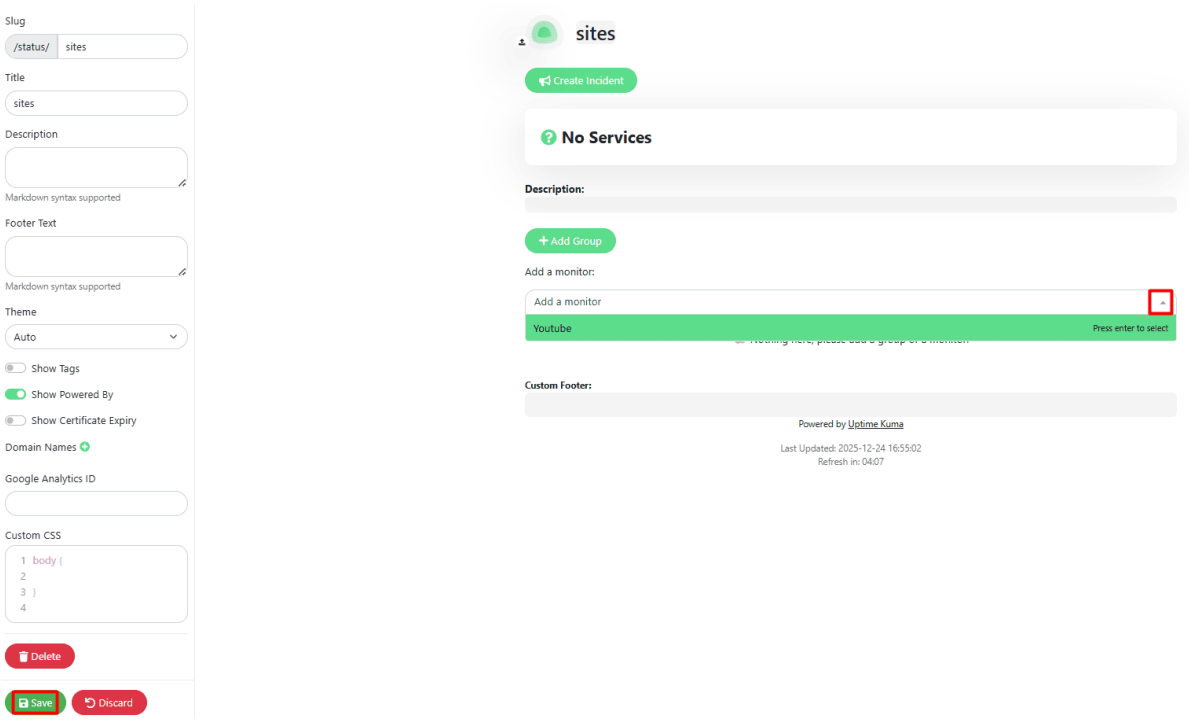
Click the New Status Page button

Click the **New Status** page button, then there will be a display as below:



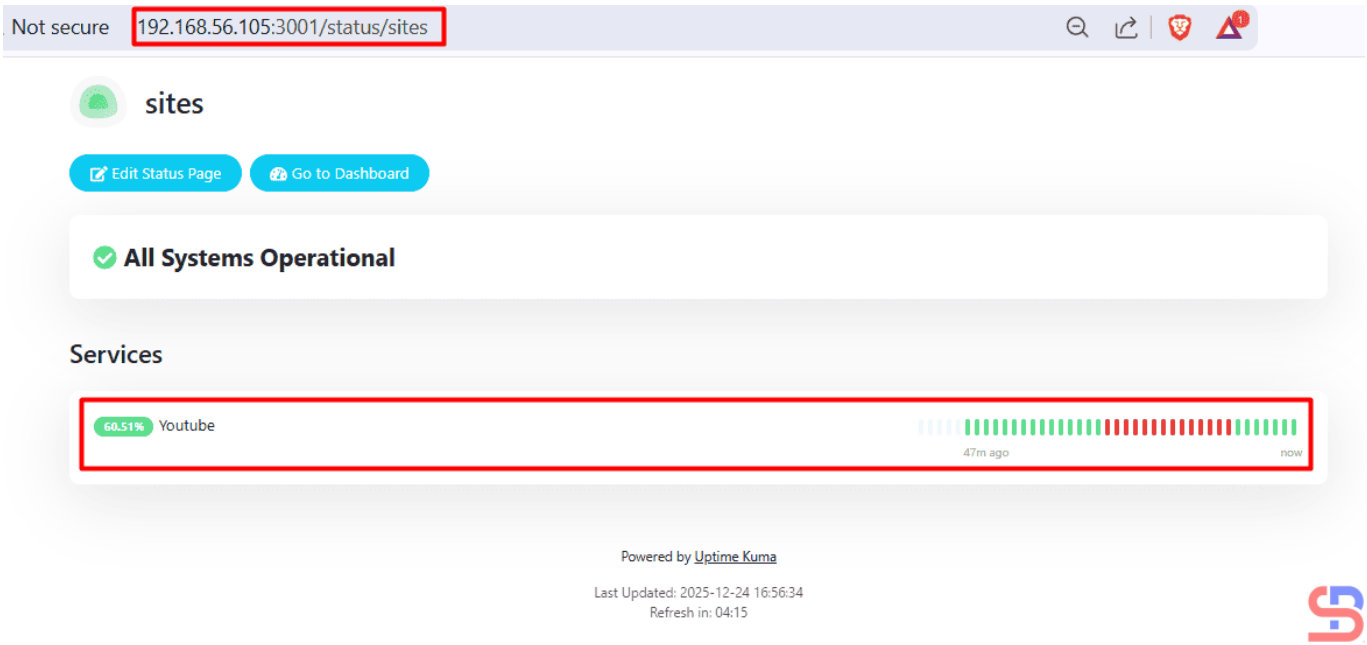
Create the Status Page page

Enter the name and slug you want (I wrote the **sites** for the name and slug), then press the **Next** button, then there will be a display as below:



Insert the host or the monitor in the Status Page

Enter the host you want to display on the Status Page, after that click the **Save** button, then there will be a display as below:



Display of Status Page

You can see that the hosts to be monitored look simpler and you can give the URL to other parties to also monitor these hosts.

Note

If you want to backup the uptime kuma database running on docker and how to restore the database, you can go to this page.

References

uptimekuma.org
kb.biznetgio.com

How to Protect phpMyAdmin Using Nginx?

written by sysadmin | 17 January 2026

The previous article explained how to install phpMyAdmin with the nginx web server. This article will explain how to protect phpMyAdmin from unauthorized users using nginx.

Problem

How to protect phpMyAdmin using nginx?

Solution

There are several methods to protect phpMyAdmin using nginx:

1. Allowing certain IPs

The phpMyAdmin application can only be accessed by users who have certain IP addresses. For example, you want the IP localhost, and only 192.168.56.1 to be able to access phpMyAdmin. Then add the script below to the **/etc/nginx/sites-available/default** file in the **location /phpmyadmin** section:

```
allow 127.0.0.1;
allow 192.168.56.1;
deny all;
```

For more details, take a look at the image below:

```
location /phpmyadmin {
    root /usr/share/;
    index index.php;

    allow 127.0.0.1;
    allow 192.168.56.1;
    deny all;

    location ~ ^/phpmyadmin/(.+\.php)$ {
        try_files $uri =404;
        root /usr/share/;
        fastcgi_pass unix:/run/php/php8.3-fpm.sock;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }

    location ~* ^/phpmyadmin/(.+\.(css|js|jpg|jpeg|gif|png|ico|html|xml|txt))$ {
        root /usr/share/;
    }
}
```

Allowing certain IPS

After that, use the command below to reload nginx:

```
sudo nginx -t
sudo systemctl reload nginx
```

If any user who uses an IP other than the localhost and 192.168.56.1 wants to access phpMyAdmin, then that user will not be able to access phpMyAdmin, as shown in the image below:



Forbidden access

2. Add a password

To make it safer, phpMyAdmin should be given additional HTTP Auth so that users who want to access the application must enter a password. Use the command below to install HTTP auth:

```
sudo apt install apache2-utils
sudo htpasswd -c /etc/nginx/.phpmyadmin admin
```

Enter the password that you want, and then in the **/etc/nginx/sites-available/default** file, add the script below:

```
auth_basic 'Restricted';
auth_basic_user_file /etc/nginx/.phpmyadmin;
```

So the default file will look like the image below:

```
location /phpmyadmin {
    root /usr/share/;
    index index.php;

    allow 127.0.0.1;
    allow 192.168.56.1;
    deny all;

    auth_basic "Restricted";
    auth_basic_user_file /etc/nginx/.phpmyadmin;


    location ~ ^/phpmyadmin/(.+\.php)$ {
        try_files $uri =404;
        root /usr/share/;
        fastcgi_pass unix:/run/php/php8.3-fpm.sock;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }

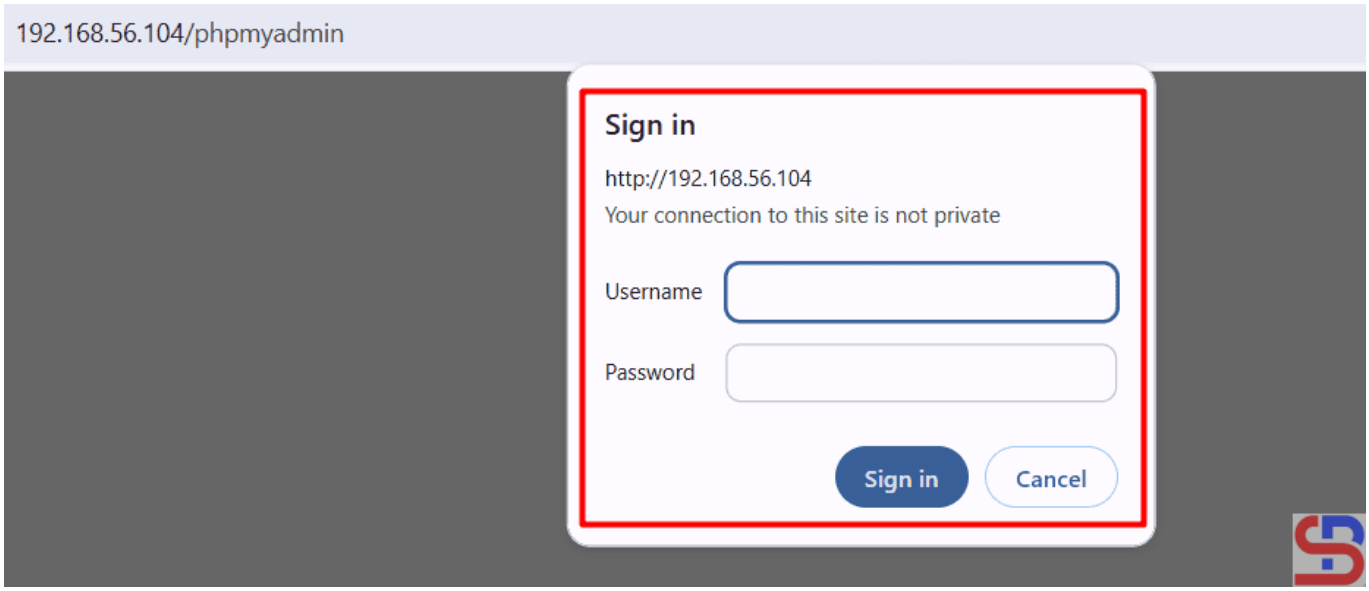
    location ~* ^/phpmyadmin/(.+\.(css|js|jpg|jpeg|gif|png|ico|html|xml|txt))$ {
        root /usr/share/;
    }
}
```

Adding HTTP Auth in Nginx

After that, use the command below to reload nginx:

```
sudo nginx -t
sudo systemctl reload nginx
```

Open the browser, and when you access phpMyAdmin, it should be there should be a display like below:



Enter username and password when accessing phpMyAdmin

Enter the username: **admin** and the password you created previously. If there are no errors, you can access phpMyAdmin.

3. Change the URL

By default, if you want to access phpMyAdmin, then you type the command below:

```
http://ip_server/phpmyadmin
```

However, for security reasons, it is best to replace the word phpMyAdmin with another word, for example, **pma,** so that the site address changes to:

```
http://ip_server/pma
```

Therefore, in the default file, change the file by deleting the **/phpmyadmin** section with the script below:

```
location /pma {
        alias /usr/share/phpmyadmin/;
        index index.php;

        allow 127.0.0.1;
```

```
        allow 192.168.56.1;
        deny all;

        auth_basic "Restricted";
        auth_basic_user_file /etc/nginx/.phpmyadmin;

        location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/run/php/php8.3-fpm.sock;
        fastcgi_param SCRIPT_FILENAME $request_filename;
        }

        location ~* \.(css|js|jpg|jpeg|gif|png|ico|html|xml|txt)$ {
        expires 30d;
        access_log off;
        }
    }
```

so that the default file changes to look like the image below:

```
server {
    listen 80;
    server_name _;
    root /var/www/html;
    index index.php index.html;

    location / {
        try_files $uri $uri/ =404;
    }

    location /pma {
        alias /usr/share/phpmyadmin/;
        index index.php;

        allow 127.0.0.1;
        allow 192.168.56.1;
        deny all;

        auth_basic "Restricted";
        auth_basic_user_file /etc/nginx/.phpmyadmin;

        location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/run/php/php8.3-fpm.sock;
        fastcgi_param SCRIPT_FILENAME $request_filename;
        }

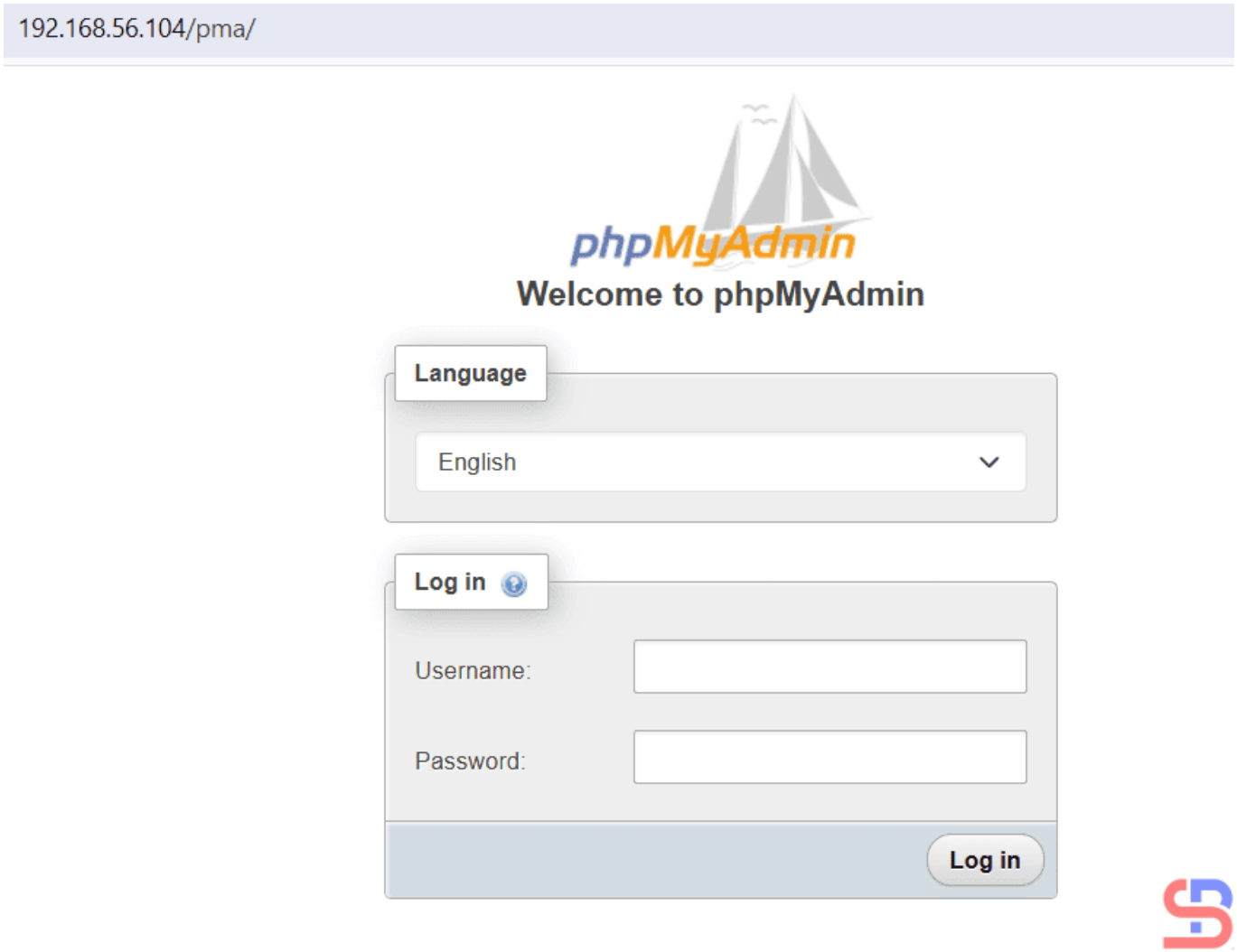
        location ~* \.(css|js|jpg|jpeg|gif|png|ico|html|xml|txt)$ {
        expires 30d;
        access_log off;
        }
    }
}
```

Change the URL in Nginx

Use the command below to reload nginx:

```
sudo nginx -t
sudo systemctl reload nginx
```

Open **http://ip_server/pma** in your browser, then you should be able to access phpMyAdmin as in the image below:



Change the URL

Note

There is one more method so that your phpMyAdmin application can be secure, namely, using SSL. You can use a Let’s Encrypt SSL certificate for your phpMyAdmin site because the certificate is free. However, if you want the phpmyadmin application not to be accessed by the public, I think, then there is no need to use SSL.

References

digitalocean.com
serverfault.com
httpd.apache.org