AI-03042 Wikijs Setup Ubuntu Server

Owner	Ⓡ Rigel Arcayan
Tags	Project
Created time	@September 28, 2025 10:42 PM

Create a step by step Wikijs Deployment on Ubuntu Server bare metal setup

Deploying Wiki.js on a bare-metal Ubuntu server involves a series of steps, from preparing the server environment to configuring the application itself. This guide outlines a common approach using **Node.js**, a **database**, and a **reverse proxy**.

---

1. Server Prerequisites and Updates 💻

Start with a fresh Ubuntu Server installation. Before you begin, ensure your system is up to date.

- **Update and Upgrade**: Run the following commands to update your package lists and upgrade all installed packages:Bash

  ```
  sudo apt update
  sudo apt upgrade -y
  ```

---

2. Install Node.js and NPM 📦

Wiki.js is built on Node.js, so you need to install the latest LTS (Long-Term Support) version. It is recommended to install it from the NodeSource repository rather than the default Ubuntu repositories to get the most recent version.

- **Add NodeSource Repository**: Use `curl` to fetch and run the setup script for the latest Node.js LTS version.Bash

  ```
  curl -fsSL https://deb.nodesource.com/setup_lts.x | sudo -E bash -
  ```

- **Install Node.js**: Install Node.js with the package manager. This command also installs npm (Node Package Manager).Bash

  ```
  sudo apt install nodejs -y
  ```

- **Verify Installation**: Confirm that Node.js and npm are installed correctly by checking their versions.Bash

  ```
  node -v
  npm -v
  ```

---

3. Install and Configure a Database 💾

Wiki.js requires a database. While it supports several, **PostgreSQL** and **MariaDB** are popular and well-documented choices for a bare-metal setup.

- **Install MariaDB**:Bash

  ```
  sudo apt install mariadb-server -y
  ```

- **Secure MariaDB**: Run the security script to set a root password and remove insecure default settings.Bash

  ```
  sudo mysql_secure_installation
  ```

- **Create Wiki.js Database and User**: Log in to the MariaDB shell and create a dedicated database and user for Wiki.js.Bash

  ```
  sudo mysql
  CREATE DATABASE wikijs;
  CREATE USER 'wikijs'@'localhost' IDENTIFIED BY 'your_strong_password';
  GRANT ALL PRIVILEGES ON wikijs.* TO 'wikijs'@'localhost';
  FLUSH PRIVILEGES;
  EXIT;
  ```

  ⚠️ **Note**: Be sure to replace `your_strong_password` with a secure password.

---

4. Deploy Wiki.js 🚀

Now, you will create a dedicated user and directory for Wiki.js and download the application files.

- **Create Dedicated User**: It's a best practice to run web applications under a non-root user for security.Bash

  ```
  sudo useradd -m -d /opt/wikijs -U -r -s /bin/bash wikijs
  ```

- **Switch to the New User**:Bash

  ```
  sudo su - wikijs
  ```

- **Download Wiki.js**: Download the latest release from the official GitHub repository.Bash

  ```
  wget https://github.com/Requarks/wiki/releases/latest/download/wiki-js.tar.gz
  ```

- **Extract and Configure**: Extract the files and copy the sample configuration.Bash

  ```
  tar -xzvf wiki-js.tar.gz
  cp config.sample.yml config.yml
  ```

- **Edit Configuration**: Open the `config.yml` file and update the database settings with the credentials you created earlier.BashYAML

  ```
  nano config.yml
  ```

  Modify the `db` section to match your database type and credentials:

  ```
  db:
    type: mariadb
    host: localhost
    port: 3306
    user: wikijs
    pass: your_strong_password
    db: wikijs
  ```

  Save and exit the editor (`Ctrl+X`, then `Y`, then `Enter`).

---

5. Create a Systemd Service ⚙️

Creating a Systemd service allows you to manage Wiki.js as a background process that automatically starts on boot.

- **Exit the `wikijs` user session**:Bash

  ```
  exit
  ```

- **Create Service File**: Create a new service file using a text editor.Bash

  ```
  sudo nano /etc/systemd/system/wikijs.service
  ```

- **Add Service Configuration**: Paste the following content into the file.1Ini, TOML

  ```
  [Unit]
  Description=Wiki.js
  After=network.target

  [Service]
  Type=simple
  ExecStart=/usr/bin/node server
  Restart=always
  User=wikijs
  ```

```
Environment=NODE_ENV=production
WorkingDirectory=/opt/wikijs

[Install]
WantedBy=multi-user.target
```

Save and exit the file.

- **Reload and Start Service**: Reload the daemon, enable the service, and start it.Bash

```
sudo systemctl daemon-reload
sudo systemctl enable wikijs
sudo systemctl start wikijs
```

- **Check Status**: Verify that the service is running.Bash

```
sudo systemctl status wikijs
```

---

6. Configure Nginx as a Reverse Proxy (Optional but Recommended) 🌐

While Wiki.js runs on its own, it's a good idea to use a reverse proxy like **Nginx** for better security, performance, and to handle SSL certificates.

- **Install Nginx**:Bash

```
sudo apt install nginx -y
```

- **Create Nginx Configuration**: Create a new server block file for Wiki.js. Replace `your_domain.com` with your actual domain name.Bash

```
sudo nano /etc/nginx/conf.d/wiki.conf
```

- **Add Nginx Configuration**:Nginx

```
server {
  listen 80;
  server_name your_domain.com;
  root /opt/wikijs;

  location / {
      proxy_pass http://127.0.0.1:3000;
      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;
      proxy_http_version 1.1;
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection "upgrade";
```

```
    }
}
```

Save and exit.

- **Test and Restart Nginx**:Bash

```
sudo nginx -t
sudo systemctl restart nginx
```

---

7. Access Wiki.js 🖥️

Finally, open your web browser and navigate to your server's IP address or the domain name you configured. You will be greeted by the Wiki.js setup wizard where you can complete the installation by providing an admin email and password.

How to Install Wiki.js on Ubuntu 22.04 | VPS Tutorial

This video provides a visual walkthrough of installing Wiki.js on an Ubuntu server, which is relevant to a bare-metal setup.

Create a step by step Wikijs Docker Deployment including all the needed services to run the application such as database and reverse proxy.

Here's a step-by-step guide to deploying Wiki.js using Docker, including a database and a reverse proxy, all managed with a single `docker-compose.yml` file. This method simplifies the setup, making it easy to manage, back up, and scale your application.

1\. Prerequisites 🛠️

First, ensure you have **Docker** and **Docker Compose** installed on your server.

- **Install Docker**: Follow the official Docker documentation for your specific operating system to install the Docker Engine.
- **Install Docker Compose**: Docker Compose is a tool for defining and running multi-container Docker applications. It's usually included with Docker Desktop but may require a separate installation on a Linux server.

2\. The Docker Compose File 📝

The core of this deployment is a `docker-compose.yml` file that defines all the services (containers), networks, and volumes. We'll set up three services:

1. `db`: A **PostgreSQL** or **MariaDB** database to store Wiki.js data.
2. `wiki`: The **Wiki.js** application itself.
3. `nginx-proxy`: An **Nginx** reverse proxy to handle web traffic and SSL.

Create a new directory for your Wiki.js project and create a file named `docker-compose.yml` inside it.

```
version: "3"

services:
  # The Database Service
  db:
    image: postgres:15-alpine
    environment:
      POSTGRES_DB: wiki
      POSTGRES_USER: wikijs
      POSTGRES_PASSWORD: your_strong_password
    volumes:
      - db-data:/var/lib/postgresql/data
    restart: unless-stopped

  # The Wiki.js Application Service
  wiki:
    image: ghcr.io/requarks/wiki:2
    environment:
      DB_TYPE: postgres
      DB_HOST: db
      DB_PORT: 5432
      DB_USER: wikijs
      DB_PASS: your_strong_password
      DB_NAME: wiki
    ports:
      - "3000:3000"
    depends_on:
      - db
```

```
    restart: unless-stopped

  # The Nginx Reverse Proxy Service
  nginx:
    image: nginx:stable-alpine
    container_name: nginx
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx.conf:/etc/nginx/conf.d/default.conf:ro
    depends_on:
      - wiki
    restart: always

# Volumes for persistent data
volumes:
  db-data:
```

- `db` **service**: We're using the official `postgres:15-alpine` image. The `environment` variables set the database name, user, and a strong password. The `volumes` section creates a named volume `db-data` to ensure your database files persist even if the container is recreated.
- `wiki` **service**: This service uses the official Wiki.js Docker image. The `environment` variables link it to the database service (`DB_HOST: db` refers to the `db` container's service name). We expose port 3000, which is the default port Wiki.js runs on.
- `nginx` **service**: This container acts as a reverse proxy. It listens on standard HTTP (port 80) and HTTPS (port 443) and forwards traffic to the Wiki.js container. We will create a custom `nginx.conf` file to configure this.

⚠️ **Note**: Be sure to replace `your_strong_password` with a secure, unique password.

3\. Configure the Nginx Reverse Proxy 🌐

Create a new file named `nginx.conf` in the same directory as your `docker-compose.yml` file. This file will instruct Nginx how to route traffic to the Wiki.js container.

```
server {
    listen 80;
    server_name your_domain.com;

    location / {
        proxy_pass <http://wiki:3000>;
        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;
    }
}
```

- `server_name`: Replace `your_domain.com` with the domain name you want to use for your wiki.
- `proxy_pass`: `http://wiki:3000` tells Nginx to forward requests to the `wiki` container on port 3000. This works because Docker Compose creates a network where containers can communicate using their service names.

4\. Deploy and Run the Containers 🚀

With the `docker-compose.yml` and `nginx.conf` files in place, you can start the entire stack with a single command.

- **Navigate to your project directory**:

  ```
  cd /path/to/your/wikijs/project
  ```

- **Run Docker Compose**:

  ```
  docker compose up -d
  ```

This command will download the necessary Docker images, create the services, and start them in the background (`-d`).

- **Check the status**: You can verify that all containers are running by using this command:

  ```
  docker compose ps
  ```

  You should see `db`, `wiki`, and `nginx` listed as "running" with "healthy" status.

5\. Access and Finalize Setup 🖥️

- Open your web browser and navigate to the IP address of your server or the domain name you configured.
- The Wiki.js setup wizard will appear. Follow the on-screen instructions to create your administrator account and complete the initial setup.

6\. Adding HTTPS with Let's Encrypt (Recommended) 🔒

To secure your wiki with SSL, you can easily integrate a service like **Certbot** or use a dedicated reverse proxy like **Nginx Proxy Manager**.

- **Using Nginx Proxy Manager**: This is a great solution that provides a web-based interface for managing reverse proxies and SSL certificates. Simply add the `nginx-proxy-manager` service to your `docker-compose.yml` file, link it to your Wiki.js container's network, and configure it through its UI. This is often the easiest path for those who aren't comfortable with manual Nginx and Certbot configurations.
- **Manual Certbot Setup**: If you prefer, you can use the Certbot Docker image to generate and renew SSL certificates.
  - Stop the `nginx` container.
  - Create a volume for your SSL certificates.
  - Run the Certbot container to get your certificates.
  - Modify your `nginx.conf` to include the SSL configuration and paths to your new certificates.

- Restart the `nginx` container.