Hello, this is the official documentation on wikijs website creation using docker image and container.

Simply pulling lscr.io/linuxserver/wikijs:latest should retrieve the correct image for your arch, but you can also pull specific arch images via tags.

The architectures supported by this image are:

Architecture	Available	Tag
x86-64	✓	amd64-<version tag>
arm64	✓	arm64v8-<version tag>
armhf	✓	arm32v7-<version tag>

Application Setup

Please note that the database configuration environment variables will apply *on first run only*, after which you will need to directly edit /config/config.yml to change your settings.

For more information, please see the official documentation.

Usage
Here are some example snippets to help you get started creating a container.

docker-compose (recommended, click here for more info)

```
---
version: "3.1"
services:
  wikijs:
    image: lscr.io/linuxserver/wikijs:latest
    container_name: wikijs
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Etc/UTC
      - DB_TYPE=sqlite #optional
      - DB_HOST= #optional
      - DB_PORT= #optional
      - DB_NAME= #optional
      - DB_USER= #optional
```

```
    - DB_PASS= #optional
  volumes:
    - /path/to/config:/config
    - /path/to/data:/data
  ports:
    - 3000:3000
  restart: unless-stopped
```

docker cli (click here for more info)

```
docker run -d \
  --name=wikijs \
  -e PUID=1000 \
  -e PGID=1000 \
  -e TZ=Etc/UTC \
  -e DB_TYPE=sqlite `#optional` \
  -e DB_HOST= `#optional` \
  -e DB_PORT= `#optional` \
  -e DB_NAME= `#optional` \
  -e DB_USER= `#optional` \
  -e DB_PASS= `#optional` \
  -p 3000:3000 \
  -v /path/to/config:/config \
  -v /path/to/data:/data \
  --restart unless-stopped \
  lscr.io/linuxserver/wikijs:latest
```

Parameters

Container images are configured using parameters passed at runtime (such as those above). These parameters are separated by a colon and indicate <external>:<internal> respectively. For example, -p 8080:80 would expose port 80 from inside the container to be accessible from the host's IP on port 8080 outside the container.

Parameter	Function
-p 3000	Port for Wiki.js's web interface.
-e PUID=1000	for UserID - see below for explanation
-e PGID=1000	for GroupID - see below for explanation

-e TZ=Etc/UTC	specify a timezone to use, see this list.
-e DB_TYPE=sqlite	Set to sqlite (default) or postgres depending on the database engine you wish to use
-e DB_HOST=	DB hostname (postgres only)
-e DB_PORT=	DB port (postgres only)
-e DB_NAME=	DB name (postgres only)
-e DB_USER=	DB username (postgres only)
-e DB_PASS=	DB password (postgres only)
-v /config	Where Wiki.js config is stored.
-v /data	Where Wiki.js data is stored.

Environment variables from files (Docker secrets)

You can set any environment variable from a file by using a special prepend FILE__.

As an example:

-e FILE__PASSWORD=/run/secrets/mysecretpassword

Will set the environment variable PASSWORD based on the contents of the /run/secrets/mysecretpassword file.

User / Group Identifiers

When using volumes (-v flags) permissions issues can arise between the host OS and the container, we avoid this issue by allowing you to specify the user PUID and group PGID.

Ensure any volume directories on the host are owned by the same user you specify and any permissions issues will vanish like magic.

In this instance PUID=1000 and PGID=1000, to find yours use id user as below:

uid=1000(dockeruser) gid=1000(dockergroup) groups=1000(dockergroup)

Docker Mods

We publish various Docker Mods to enable additional functionality within the containers. The list of Mods available for this image (if any) as well as universal mods that can be applied to any one of our images can be accessed via the dynamic badges above.

Support Info

- **Shell access whilst the container is running: docker exec -it wikijs /bin/bash**
- **To monitor the logs of the container in realtime: docker logs -f wikijs**
- **container version number**
    - **docker inspect -f '{{ index .Config.Labels "build_version" }}' wikijs**
- **image version number**
    - **docker inspect -f '{{ index .Config.Labels "build_version" }}' lscr.io/linuxserver/wikijs:latest**

Updating Info

Most of our images are static, versioned, and require an image update and container recreation to update the app inside. With some exceptions (ie. nextcloud, plex), we do not recommend or support updating apps inside the container. Please consult the Application Setup section above to see if it is recommended for the image.
Below are the instructions for updating containers:

Via Docker Compose

- **Update all images: docker-compose pull**
    - **or update a single image: docker-compose pull wikijs**
- **Let compose update all containers as necessary: docker-compose up -d**
    - **or update a single container: docker-compose up -d wikijs**
- **You can also remove the old dangling images: docker image prune**

Via Docker Run

- **Update the image: docker pull lscr.io/linuxserver/wikijs:latest**
- **Stop the running container: docker stop wikijs**
- **Delete the container: docker rm wikijs**
- **Recreate a new container with the same docker run parameters as instructed above (if mapped correctly to a host folder, your /config folder and settings will be preserved)**

- You can also remove the old dangling images: docker image prune

Via Watchtower auto-updater (only use if you don't remember the original parameters)

- Pull the latest image at its tag and replace it with the same env variables in one run:
- docker run --rm \
  -v /var/run/docker.sock:/var/run/docker.sock \
  containrrr/watchtower \
  --run-once wikijs

- You can also remove the old dangling images: docker image prune

Note: We do not endorse the use of Watchtower as a solution to automated updates of existing Docker containers. In fact we generally discourage automated updates. However, this is a useful tool for one-time manual updates of containers where you have forgotten the original parameters. In the long term, we highly recommend using Docker Compose.

Image Update Notifications - Diun (Docker Image Update Notifier)

- We recommend Diun for update notifications. Other tools that automatically update containers unattended are not recommended or supported.

Building locally

If you want to make local modifications to these images for development purposes or just to customize the logic:
git clone https://github.com/linuxserver/docker-wikijs.git
cd docker-wikijs
docker build \
 --no-cache \
 --pull \
 -t lscr.io/linuxserver/wikijs:latest .

The ARM variants can be built on x86_64 hardware using multiarch/qemu-user-static

docker run --rm --privileged multiarch/qemu-user-static:register --reset
Once registered you can define the dockerfile to use with -f Dockerfile.aarch64.