

# Docker Compose

The Compose file provides a way to document and configure all of the application's service dependencies (databases, queues, caches, web service APIs, etc). Using the Compose command line tool you can create and start one or more containers for each dependency with a single command.

- [Portainer 2.0 Update](#)
- [Install Snapdrop using Compose](#)
- [Install Droppy with Docker Compose](#)
- [Backup Docker Containers using Resilio Sync](#)
- [DashMachine 0.7 with Code Server](#)
- [Install Wordpress using Docker Compose](#)
- [Monica HQ](#)
- [OliveTin Linux Shell Web Interface](#)
- [Setup ShareX to work with Xbackbone from Windows](#)
- [Setup Vikunja using Docker Compose](#)
- [Setup and Install Kiwix Serve on Debian Systems](#)
- [Install Memo Notes via Docker Compose](#)
- [Install Polr Link Shortener with Docker Compose](#)

# Portainer 2.0 Update

Use this compose to update to Portainer 2.0

```
version: '2'

services:
  portainer:
    image: portainer/portainer-ce
    command: -H unix:///var/run/docker.sock
    restart: always
    ports:
      - 9001:9000
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - portainer_data:/data
```

# Install Snapdrop using Compose

Use this compose stack to install Snapdrop. Change ports as needed. More information can be found [here](#).

```
---
version: "2.1"
services:
  snapdrop:
    image: linuxserver/snapdrop
    container_name: snapdrop
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Europe/London
    ports:
      - 80:80
      - 443:443
    restart: unless-stopped
```

## Install using Portainer

1. Click on Stacks
2. Give the stack a name
3. Paste the compose script above into the stacks editor on Portainer
4. Make any changes to ports (if necessary)
5. Deploy

portainer.io

admin

my account log out

### Create stack

Stacks > Add stack

Name:

This stack will be deployed using the equivalent of `docker-compose`. Only Compose file format version 2 is supported at the moment.

Note: Due to a limitation of libcompose, the name of the stack will be standardized to remove all special characters and uppercase letters.

Build method

- Web editor  
Use our Web editor
- Upload  
Upload from your computer
- Repository  
Use a git repository
- Custom template  
Use a custom template

Web editor

You can get more information about Compose file format in the [official documentation](#).

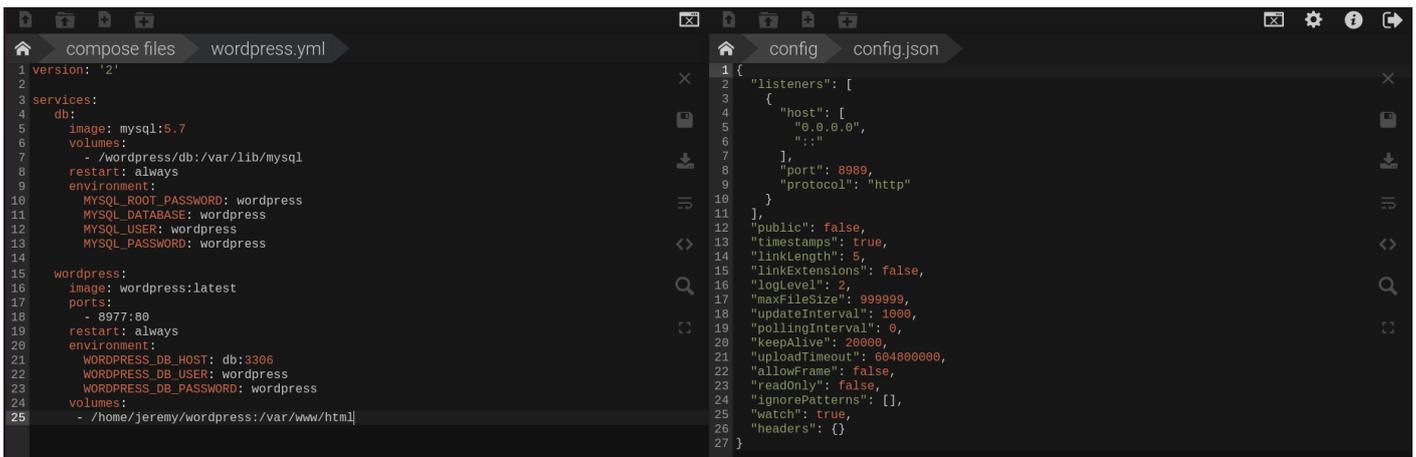
```
1 ---
2 version: "2.1"
3 services:
4   snapdrop:
5     image: linuxserver/snapdrop
6     container_name: snapdrop
7     environment:
8       - PUID=1000
9       - PGID=1000
10      - TZ=Europe/London
11    ports:
12      - 80:80
13      - 443:443
14    restart: unless-stopped
```

# What is Snapdrop?

<https://www.youtube.com/embed/5BHI9TRshpE>

# Install Droppy with Docker Compose

**Droppy** is a self-hosted file storage server with a web interface and capabilities to edit files and view media directly in the browser. It is particularly well-suited to be run on low-end hardware like the Raspberry Pi.



The image shows a terminal window with two files open. The left file is 'wordpress.yml' and the right file is 'config.json'. The 'wordpress.yml' file contains a Docker Compose configuration for a WordPress service and a MySQL database service. The 'config.json' file contains a configuration for the Droppy web interface, including listeners, public status, timestamps, link length, log level, max file size, update interval, polling interval, keep alive, upload timeout, allow frame, read only, ignore patterns, watch, and headers.

```
1 version: '2'
2
3 services:
4   db:
5     image: mysql:5.7
6     volumes:
7       - /wordpress/db:/var/lib/mysql
8     restart: always
9     environment:
10      MYSQL_ROOT_PASSWORD: wordpress
11      MYSQL_DATABASE: wordpress
12      MYSQL_USER: wordpress
13      MYSQL_PASSWORD: wordpress
14
15   wordpress:
16     image: wordpress:latest
17     ports:
18       - 8977:80
19     restart: always
20     environment:
21       WORDPRESS_DB_HOST: db:3306
22       WORDPRESS_DB_USER: wordpress
23       WORDPRESS_DB_PASSWORD: wordpress
24     volumes:
25       - /home/jeremy/wordpress:/var/www/html
```

```
1 {
2   "listeners": [
3     {
4       "host": [
5         "0.0.0.0",
6         "::"
7       ],
8       "port": 8989,
9       "protocol": "http"
10    }
11  ],
12  "public": false,
13  "timestamps": true,
14  "linkLength": 5,
15  "linkExtensions": false,
16  "logLevel": 2,
17  "maxFileSize": 999999,
18  "updateInterval": 1000,
19  "pollingInterval": 0,
20  "keepAlive": 20000,
21  "uploadTimeout": 604800000,
22  "allowFrame": false,
23  "readOnly": false,
24  "ignorePatterns": [],
25  "watch": true,
26  "headers": {}
27 }
```

## Features

- Responsive, scalable HTML5 interface
- Realtime updates of file system changes
- Directory and Multi-File upload
- Drag-and-Drop support
- Clipboard support to create image/text files
- Side-by-Side mode
- Simple and fast Search
- Shareable public download links
- Zip download of directories
- Powerful text editor with themes and broad language support
- Image and video gallery with touch support
- Audio player with seeking support
- Fullscreen support for editor and gallery
- Supports installing to the homescreen
- Docker images available for x86-64, ARMv6, ARMv7 and ARMv8

Install Droppy file manager using this compose stack. Change ports or volumes as needed. See the [Droppy GitHub repo](#) for more details.

```
version: '2'
services:
  droppy:
    container_name: droppy
    image: silverwind/droppy
    ports:
      - 8989:8989
    volumes:
      - /docker/Droppy:/config
      - /docker/Droppy/files:/files
    restart: unless-stopped
```

[https://www.youtube.com/embed/8e-YAxQZ\\_NA](https://www.youtube.com/embed/8e-YAxQZ_NA)

# Backup Docker Containers using Resilio Sync

Resilio Sync is a proprietary peer-to-peer file synchronization tool available for Windows, Mac, Linux, Android, iOS, Windows Phone, Amazon Kindle Fire and BSD.

Resilio Sync can be used to copy your docker container files from one machine to another either locally or anywhere in the world. You will need to install Resilio Sync on both machines. My Docker host is on a VM and the backups are sent to my Synology NAS which also runs Resilio Sync. For Synology users, Resilio Sync can be installed as a Package or as a Docker container. I prefer the Docker container but both will work. For the Docker host machine I only use... well, the docker container of-course.

One thing to note is Resilio Sync is NOT a backup tool but a syncing tool. Whatever your host machine pulls in, your backup machine will too. Keep that in mind when it comes to downloading files that you have no idea might contain threats to your system.

## Install Resilio Sync

Install the LinuxServer Resilio Sync Docker container on your Docker host. I use this compose file.

```
---
version: "2.1"
services:
  resilio-sync:
    image: linuxserver/resilio-sync
    container_name: resilio-sync
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Europe/London
      - UMASK_SET=022
    volumes:
      - path to config: /config
      - path to downloads: /downloads
```

```
- /: /sync
ports:
- 8888: 8888
- 55555: 55555
restart: unless-stopped
```

I prefer Compose because I use Portainer and Yacht to control my stacks.

Make changes to the ports, IDs and volumes as needed. Notice I bind "/" to "/sync", this allows you to browse your whole machine for folders to sync.

Repeat the same process on your Docker Backup machine.

## Setup Resilio Sync

When you first load the UI you will be prompted to setup a username and password. Then assign a machine name. This name will be used so you can identify which machine is syncing to the folders you share. It's good if you have multiple machines syncing files.

Now on your Docker host Resilio Sync instance, you need to add a folder to share out. This should be the folder where you store your Docker container/s.

Click the plus in the upper left and add a standard folder.

My container files happen to be in my root directory. This is where my Wordpress files are stored for my Wordpress container.

Select the folder then click open. You will get a prompt to share your folder. Close that because we are not ready to do that yet.

Before we share anything we need to go to our backup machine and create a folder where the files will be synced. In my case, this is my Synology NAS. I created a new share and this is where all my Docker backups will sync to.

I created a new folder called Family Portal and that is where I want the files from my Docker host to go to. Now it's time to sync the files from your host to your backup machine.

Go back to your Docker host Resilio Sync instance and right click on the folder you just setup and click the share option.

This will give you some different ways to sync your files to the backup machine.

I like to do "read only" syncing because this will make it so the backup machine cannot delete files from the host machine. So if you accidentally delete the files on your backup machine it won't delete them on the host machine too! Once you've made that mistake it only takes one time before

you use read only lol.

Click on Key and copy the Read Only link then go back to your backup machine instance of Resilio Sync.

Click on the plus icon in the upper left corner again and this time click on Enter a key or link.

A new window will pop up where you enter the key and press ok. Then you need to browse for the folder you made on your backup machine to sync the files. I made the Family Portal folder so that's the folder I chose. Then click open.

Your files should now begin downloading to the backup machine and will do so every time a new file is added or changed.

Resilio Sync is very low maintenance. It's pretty much set it and forget it. However, it's always good to peek in and make sure your connections are still working from time to time.

# DashMachine 0.7 with Code Server

```
---
version: "2.1"
services:
  dashmachine:
    image: rmountjoy/dashmachine0.7:latest
    container_name: dashmachine
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=America/Detroit
      - EDITOR_URL=https://your-reverse-proxy.url
    volumes:
      - /config/directory:/DashMachine/config
    ports:
      - 5000:5000
    restart: unless-stopped
  code-server:
    image: linuxserver/code-server:latest
    container_name: code-server
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=America/Detroit
      - PASSWORD=mysecretpassword
      - SUDO_PASSWORD=mysecretpassword
      - PROXY_DOMAIN=your-reverse-proxy.url
    volumes:
      - /config/directory:/config
      - /config/directory/workspace:/config/workspace/Dashmachine
    ports:
      - 8443:8443
    restart: unless-stopped
```



# Install Wordpress using Docker Compose

This stack can be used to install Wordpress with a MySQL DB. Change the ports and volume binds as you need.

```
version: '2'

services:
  db:
    image: mysql:5.7
    volumes:
      - /docker/wordpress/db:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: wordpress
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress

  wordpress:
    image: wordpress:latest
    ports:
      - 8977:80
    restart: always
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
    volumes:
      - /docker/wordpress:/var/www/html
```

# Monica HQ

This version will use the apache image and add a mysql container. The volumes are set to keep your data persistent. This setup provides **no ssl encryption** and is intended to run behind a proxy.

Make sure to pass in values for `APP_KEY` variable before you run this setup.

Set `APP_KEY` to a random 32-character string. You can for instance copy and paste the output of `echo -n 'base64:'; openssl rand -base64 32`.

## 1. Create a `docker-compose.yml` file

```
version: "3.4"

services:
  app:
    image: monica
    depends_on:
      - db
    ports:
      - 8080:80
    environment:
      - APP_KEY=
      - DB_HOST=db
      - DB_USERNAME=usermonica
      - DB_PASSWORD=secret
    volumes:
      - data: /var/www/html/storage
    restart: always

  db:
    image: mysql:5.7
    environment:
      - MYSQL_RANDOM_ROOT_PASSWORD=true
      - MYSQL_DATABASE=monica
      - MYSQL_USER=usermonica
      - MYSQL_PASSWORD=secret
    volumes:
```

```
- mysql: /var/lib/mysql  
restart: always
```

```
volumes:
```

```
  data:
```

```
    name: data
```

```
  mysql:
```

```
    name: mysql
```

# OliveTin Linux Shell Web Interface

## OliveTin

OliveTin is a web interface for running Linux shell commands.

 or t

[Discord](#) [Go Report Card](#)

Some example **use cases**;

1. Give controlled access to run shell commands to less technical folks who cannot be trusted with SSH. I use this so my family can `podman restart plex` without asking me, and without giving them shell access!
2. Great for home automation tablets stuck on walls around your house - I use this to turn Hue lights on and off for example.
3. Sometimes SSH access isn't possible to a server, or you are feeling too lazy to type a long command you run regularly! I use this to send Wake on Lan commands to servers around my house.

[Join the community on Discord.](#)

## YouTube video demo (6 mins)

[6 minute demo video](#)

## Features

- **Responsive, touch-friendly UI** - great for tablets and mobile
- **Super simple config in YAML** - because if it's not YAML now-a-days, it's not "cloud native" :-)
- **Dark mode** - for those of you that roll that way.

- **Accessible** - passes all the accessibility checks in Firefox, and issues with accessibility are taken seriously.
- **Container** - available for quickly testing and getting it up and running, great for the selfhosted community.
- **Integrate with anything** - OliveTin just runs Linux shell commands, so theoretically you could integrate with a bunch of stuff just by using curl, ping, etc. However, writing your own shell scripts is a great way to extend OliveTin.
- **Lightweight on resources** - uses only a few MB of RAM and barely any CPU. Written in Go, with a web interface written as a modern, responsive, Single Page App that uses the REST/gRPC API.
- **Good amount of unit tests and style checks** - helps potential contributors be consistent, and helps with maintainability.

# Screenshots

Desktop web browser;

Desktop screenshot 

Desktop web browser (dark mode);

Desktop screenshot 

Mobile screen size (responsive layout);

Mobile screenshot 

# Documentation

All documentation can be found at <http://docs.olivetin.app> . This includes installation and usage guide, etc.

## Quickstart reference for `config.yaml`

This is a quick example of `config.yaml` - but again, lots of documentation for how to write your `config.yaml` can be found at [the documentation site](#).

Put this `config.yaml` in `/etc/OliveTin/` if you're running a standard service, or mount it at `/config` if running in a container.

```
# Listen on all addresses available, port 1337
listenAddressSingleHTTPFrontend: 0.0.0.0:1337

# Choose from INFO (default), WARN and DEBUG
LogLevel: "INFO"

# Actions (buttons) to show up on the WebUI:
actions:
  # Docs: https://docs.olivetin.app/action-container-control.html
  - title: Restart Plex
    icon: smile
    shell: docker restart plex

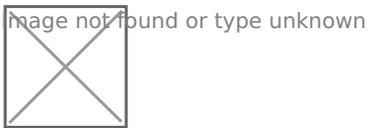
  # This will send 1 ping
  # Docs: https://docs.olivetin.app/action-ping.html
  - title: Ping Google.com
    shell: ping google.com -c 1

  # Restart lightdm on host "overseer"
  # Docs: https://docs.olivetin.app/action-ssh.html
  - title: restart lightdm
    icon: poop
    shell: ssh root@overseer 'service lightdm restart'
```

A full example config can be found at in this repository - [config.yaml](#).

# Setup ShareX to work with Xbackbone from Windows

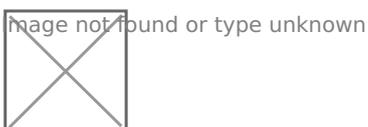
XBackBone is a simple, self-hosted, lightweight PHP file manager that support the instant sharing tool ShareX and \*NIX systems. It supports uploading and displaying images, GIF, video, code, formatted text, and file downloading and uploading. Also have a web UI with multi user management, past uploads history and search support.



Install Xbackbone server using this Docker Compose snippet.

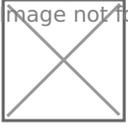
```
version: "2.1"
services:
  xbackbone:
    image: ghcr.io/linuxserver/xbackbone
    container_name: xbackbone
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=America/Detroit
    volumes:
      - /docker/xbackbone:/config
    ports:
      - 8087: 80
      - 4443: 443
    restart: unless-stopped
```

Navigate to your server IP on port 8087. I chose the SQLite option for the database. Setup your login then go to your profile page to setup the token. Press 'Update' to get your token.



Now download the Sharex client config by clicking the button shown here

image not found or type unknown



Now download [ShareX](#) on your Windows machine then right click on the Sharex icon in your system tray to go to Destinations > Custom Uploader Settings.

image not found or type unknown



Now it's time to import that file you downloaded from your Xbackbone profile.

image not found or type unknown



Then under Image uploader (2) select your domain or server IP. Click the test button to make sure it works, then close it out.

## Extra

You can enable the ability to right click on images from websites or your desktop to upload to your server.

image not found or type unknown



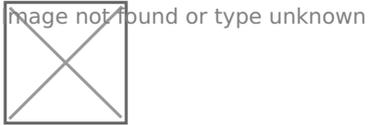
Open the ShareX Application Settings and go to Integration then make sure to tick "Show Upload with Sharex button in Windows context menu.

Now enjoy your self hosted Xbackbone host!

image not found or type unknown



# Setup Vikunja using Docker Compose



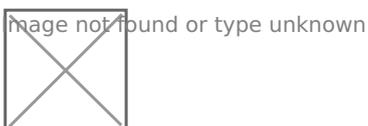
Vikunja is an Open-Source, self-hosted To-Do list application for all platforms. It is licensed under the GPLv3.

Test out [Vikunja demo here](#).

This guide is assuming you are installing Vikunja on a fresh Debian based OS such as Ubuntu, Debian or Turnkey Core. You can edit the docker-compose.yml to change the port and database passwords however, it is not required unless port 80 is being used by another service. **Do NOT modify or change anything in the nginx.conf file** even if you change the port in the docker-compose.yml file.

1. Install docker - apt install docker.io -y
2. Install docker compose - apt install docker-compose -y
3. cd /
4. mkdir docker
5. cd docker
6. mkdir vikunja
7. cd /docker/vikunja
8. create docker-compose.yml - touch docker-compose.yml
9. nano docker-compose.yml - copy the content below into the file then press ctrl x then y then enter.
10. create nginx.conf - touch nginx.conf - If that creates a directory, remove it rm -d nginx.conf then use cat > nginx.conf
11. nano nginx.conf - copy the content from the below example into the file then press ctrl x then y then enter on your keyboard.
12. run docker-compose up -d

When the install finishes you should see these files and folders in /docker/vikunja directory.



Navigate to the IP of your machine and register an account on Vikunja. There is no admin account.

Find the ip by typing the following into the terminal:

```
ip addr
```

## Vikunja Docker Compose without Email Notifications

```
version: '3'

services:
  db:
    image: mariadb:10
    command: --character-set-server=utf8mb4 --collation-server=utf8mb4_unicode_ci
    environment:
      MYSQL_ROOT_PASSWORD: supersecret
      MYSQL_USER: vikunja
      MYSQL_PASSWORD: secret
      MYSQL_DATABASE: vikunja
    volumes:
      - ./db: /var/lib/mysql
    restart: unless-stopped
  api:
    image: vikunja/api
    environment:
      VIKUNJA_DATABASE_HOST: db
      VIKUNJA_DATABASE_PASSWORD: secret
      VIKUNJA_DATABASE_TYPE: mysql
      VIKUNJA_DATABASE_USER: vikunja
      VIKUNJA_DATABASE_DATABASE: vikunja
    volumes:
      - ./files: /app/vikunja/files
    depends_on:
      - db
    restart: unless-stopped
  frontend:
    image: vikunja/frontend
    restart: unless-stopped
  proxy:
    image: nginx
```

```
ports:
  - 8022: 80
volumes:
  - ./nginx.conf:/etc/nginx/conf.d/default.conf:ro
depends_on:
  - api
  - frontend
restart: unless-stopped
```

## Vikunja Nginx.conf file

```
server {
    listen 80;

    location / {
        proxy_pass http://frontend:80;
    }

    location ~* ^/(api|dav|\.well-known)/ {
        proxy_pass http://api:3456;
        client_max_body_size 20M;
    }
}
```

## Vikunja Docker Compose With Email Notifications

The following docker compose is using Google SMTP servers to send email notifications. Edit the SMTP settings in the stack below to suit your needs.

```
version: '3'

services:
  db:
    image: mariadb:10
    command: --character-set-server=utf8mb4 --collation-server=utf8mb4_unicode_ci
    environment:
      MYSQL_ROOT_PASSWORD: supersecret
      MYSQL_USER: vikunja
      MYSQL_PASSWORD: secret
      MYSQL_DATABASE: vikunja
```

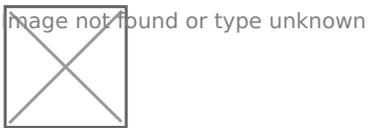
```
volumes:
  - ./db: /var/lib/mysql
restart: unless-stopped
api:
  image: vikunja/api
  environment:
    VIKUNJA_DATABASE_HOST: db
    VIKUNJA_DATABASE_PASSWORD: secret
    VIKUNJA_DATABASE_TYPE: mysql
    VIKUNJA_DATABASE_USER: vikunja
    VIKUNJA_DATABASE_DATABASE: vikunja
    VIKUNJA_SERVICE_FRONTENDURL: https://your-frontend-url.com/
    VIKUNJA_SERVICE_ENABLETASKATTACHMENTS: 1
    VIKUNJA_SERVICE_ENABLEREGISTRATION: 0
    VIKUNJA_SERVICE_ENABLEEMAILREMINDERS: 1
    VIKUNJA_MAILER_ENABLED: 1
    VIKUNJA_MAILER_FORCESSL: 1
    VIKUNJA_MAILER_HOST: smtp.gmail.com
    VIKUNJA_MAILER_PORT: 465
    VIKUNJA_MAILER_USERNAME: youremail@gmail.com
    VIKUNJA_MAILER_PASSWORD: yourgmailpassword
  volumes:
    - ./files: /app/vikunja/files
  depends_on:
    - db
  restart: unless-stopped
frontend:
  image: vikunja/frontend
  restart: unless-stopped
proxy:
  image: nginx
  ports:
    - 8022: 80
  volumes:
    - ./nginx.conf: /etc/nginx/conf.d/default.conf:ro
  depends_on:
    - api
    - frontend
  restart: unless-stopped
```

# Video Tutorials

[https://www.youtube.com/embed/fGlz2PkXjuo?ab\\_channel=Geeked](https://www.youtube.com/embed/fGlz2PkXjuo?ab_channel=Geeked)

# Setup and Install Kiwix Serve on Debian Systems

Kiwix Serve is a .zim compatible web server: it allows you to deliver .zim files over the HTTP protocol within your local network – be it a University or your own house. Simply start Kiwix-Serve on your machine, and your content will be available for anybody through their web browser.



Kiwix is an offline reader – meaning that it allows you to browse text or video that is normally only available on the internet. We turn various online contents (such as Wikipedia, for example) into ZIM files, and these can be opened by Kiwix even if you have no connectivity.

Our technology's main advantage is its high compression rate. For instance, the entirety of Wikipedia (more than 6 million articles, with images) can fit in 80Gb. The Gutenberg Library's 60,000 books will fit on 60 Gb of storage space.

## Docker Compose Stack

Change the volume location to that of your own and add the correct zim file names that are in that directory to the stack.

```
version: '3'
services:
  kiwix-serve:
    image: kiwix/kiwix-serve
    volumes:
      - /docker/kiwixserve:/data
    ports:
      - '8080:80'
    command:
      wikipedia_en_top_maxi_2021-08.zim
```

It is advised to just use the run command and use an asterisk so you don't have to add or change the zim filenames in the compose stack above. It's just easier to use.

```
docker run -v /docker/kiwixserve:/data -p 8080:80 kiwix/kiwix-serve *.zim
```

## Kiwix Files I Host

All of the Kiwix zim files I host with detailed information

wikipedia_en_all_maxi_2021-03.zim	<a href="https://download.kiwix.org/zim/wikipedia/wikipedia_en_all_maxi_2021-03.zim">https://download.kiwix.org/zim/wikipedia/wikipedia_en_all_maxi_2021-03.zim</a>	82G
askubuntu.com_en_all_2021-05.zim	<a href="https://download.kiwix.org/zim/stackexchange/askubuntu.com_en_all_2021-05.zim">https://download.kiwix.org/zim/stackexchange/askubuntu.com_en_all_2021-05.zim</a>	8.1G
gardening.stackexchange.com_en_all_2021-05.zim	<a href="https://download.kiwix.org/zim/stackexchange/gardening.stackexchange.com_en_all_2021-05.zim">https://download.kiwix.org/zim/stackexchange/gardening.stackexchange.com_en_all_2021-05.zim</a>	938M
gutenberg_en_all_2021-10.zim	<a href="https://download.kiwix.org/zim/gutenberg/gutenberg_en_all_2021-10.zim">https://download.kiwix.org/zim/gutenberg/gutenberg_en_all_2021-10.zim</a>	64G
crashcourse_en_all_2021-09.zim	<a href="https://download.kiwix.org/zim/other/crashcourse_en_all_2021-09.zim">https://download.kiwix.org/zim/other/crashcourse_en_all_2021-09.zim</a>	39G
wikibooks_en_all_maxi_2021-03.zim	<a href="https://download.kiwix.org/zim/wikibooks/wikibooks_en_all_maxi_2021-03.zim">https://download.kiwix.org/zim/wikibooks/wikibooks_en_all_maxi_2021-03.zim</a>	4.3G
wiktionary_en_all_maxi_2021-09.zim	<a href="https://download.kiwix.org/zim/wiktionary/wiktionary_en_all_maxi_2021-09.zim">https://download.kiwix.org/zim/wiktionary/wiktionary_en_all_maxi_2021-09.zim</a>	6G
superuser.com_en_all_2020-10.zim	<a href="https://download.kiwix.org/zim/stackexchange/superuser.com_en_all_2020-10.zim">https://download.kiwix.org/zim/stackexchange/superuser.com_en_all_2020-10.zim</a>	8.5G
wikisource_en_all_maxi_2021-03.zim	<a href="https://download.kiwix.org/zim/wikisource/wikisource_en_all_maxi_2021-03.zim">https://download.kiwix.org/zim/wikisource/wikisource_en_all_maxi_2021-03.zim</a>	14G
ted_en_technology_2021-10.zim	<a href="https://download.kiwix.org/zim/ted/ted_en_technology_2021-10.zim">https://download.kiwix.org/zim/ted/ted_en_technology_2021-10.zim</a>	45G
wikistage_multi_all_2021-09.zim	<a href="https://download.kiwix.org/zim/other/wikistage_multi_all_2021-09.zim">https://download.kiwix.org/zim/other/wikistage_multi_all_2021-09.zim</a>	21G
wikivoyage_en_all_maxi_2021-10.zim	<a href="https://download.kiwix.org/zim/wikivoyage/wikivoyage_en_all_maxi_2021-10.zim">https://download.kiwix.org/zim/wikivoyage/wikivoyage_en_all_maxi_2021-10.zim</a>	658M
wikinews_en_all_maxi_2021-08.zim	<a href="https://download.kiwix.org/zim/wikinews/wikinews_en_all_maxi_2021-08.zim">https://download.kiwix.org/zim/wikinews/wikinews_en_all_maxi_2021-08.zim</a>	231M
wikispecies_en_all_maxi_2021-08.zim	<a href="https://download.kiwix.org/zim/other/wikispecies_en_all_maxi_2021-08.zim">https://download.kiwix.org/zim/other/wikispecies_en_all_maxi_2021-08.zim</a>	2.2G

wikiversity_en_all_maxi_2021-03.zim	<a href="https://download.kiwix.org/zim/wikiversity/wikiversity_en_all_maxi_2021-03.zim">https://download.kiwix.org/zim/wikiversity/wikiversity_en_all_maxi_2021-03.zim</a>	2.3G
-------------------------------------	---	------

Find and download .zim files here <https://download.kiwix.org/zim/>

## What do mini, nopic and maxi mean in the Wikipedia zim files?

File size is always an issue when downloading such big content, so Kiwix produces each Wikipedia file in three flavours:

- *Mini*: only the introduction of each article, plus the infobox. Saves about 95% of space vs. the full version.
- *nopic*: full articles, but no images. About 75% smaller than the full version
- *Maxi*: the default full version.

1. CD into your mapped data directory for the Docker container to download the zim files using wget.
2. Add the name of each zim file to your stack.
3. Start or deploy the docker stack and enjoy!

There is also a desktop app for both Windows and Linux called [Kiwix JS Electron](#).

## Pause wget Download

If you find yourself in a situation where you're downloading a large file and need to shutdown your computer during the download for whatever reason, no problem, you can pause the download.

In the terminal window that is downloading your file just enter the following keyboard shortcut.

```
Ctrl + c
```

This will stop the download and then you can shutdown your computer if you need. To resume the download, read on.

## Resume wget Download

To resume a wget download it's very straight forward. Open the terminal to the directory where you were downloading your file to and run wget with the -c flag to resume the download.

```
wget -c https://example.com/filename.zip
```

# Install Memo Notes via Docker Compose

Use this guide to setup and install Meemo with Docker Compose. See our [review of Meemo](#).

## Setup

1. Ensure Docker and Docker-Compose are installed
2. On your host machine, create the following files and directories inside a directory of your choice.

```
# users file
touch users.json
# data and database directory
mkdir data database
# set ownership to map container user ID 1000
chown 1000 users.json data database
# set permissions
chmod 600 users.json
chmod 700 data database
```

3. Download [docker-compose.yml](#) on your host inside the same directory then modify it as you wish:

```
wget https://raw.githubusercontent.com/qdm12/meemo/master/docker-compose.yml
```

4. Launch the MongoDB database and Meemo container with

```
docker-compose up -d
```

5. You can check logs with

```
docker-compose logs -f
```

6. Meemo is at localhost:3000 (depending on your mapped port in docker-compose.yml)

## Configuration

We assume your Meemo container is named `meemo` in the following.

```
# List users
docker exec meemo ./meemo/admin users

# Add a user
docker exec meemo ./meemo/admin user-add -u yourUser -p yourPassword --display-name yourUser

# Edit a user
docker exec meemo ./meemo/admin user-edit -u yourUser -p yourPassword --display-name yourUser

# Remove a user
docker exec meemo ./meemo/admin user-del -u yourUser
```

All the changes are saved to `users.json`

To support the Meemo project or follow development, please visit the [Meemo Github repo](#).

# Install Polr Link Shortener with Docker Compose

*Polr is a quick, modern, and open-source link shortener. It allows you to host your own URL shortener, to brand your URLs, and to gain control over your data. [Visit the Polar website.](#)*

Image not found or type unknown



If MySQL is not already setup, run this stack to install it for Polr to connect to a DB.

```
version: '3'

services:
  db:
    image: mysql:5.7
    container_name: db
    environment:
      MYSQL_ROOT_PASSWORD: supersecretdbpass0rd
      MYSQL_DATABASE: polr
      MYSQL_USER: root
      MYSQL_PASSWORD: supersecretdbpass0rd
    ports:
      - "3306:3306"
    volumes:
      - /docker/mysql:/var/lib/mysql
  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    container_name: pma
    links:
      - db
    environment:
      PMA_HOST: db
      PMA_PORT: 3306
      PMA_ARBITRARY: 1
```

```
restart: always
ports:
  - 8099: 80
volumes:
  dbdata:
```

Use this docker stack.

```
version: '3.3'
services:
  polr:
    ports:
      - '8080: 8080'
    environment:
      - DB_HOST=192.168.1.100: 3306 #your docker host IP
      - DB_DATABASE=polr
      - DB_USERNAME=root
      - DB_PASSWORD=supersecretdbpass0rd
      - APP_ADDRESS=yourdomainhere.com
      - ADMIN_USERNAME=admin
      - ADMIN_PASSWORD=admin
      - SETTING_ADV_ANALYTICS=true
    image: ajanvier/polr
```

Visit your Polr app at [yourdomain.com](http://yourdomain.com)