### How to Host a Joplin Server with Docker on Ubuntu

Joplin is a robust note-taking app that is open-source and allows for synchronization on various devices. Running your own Joplin server gives you complete authority over your data and guarantees privacy. This manual will guide you on how to establish and manage a Joplin server on Ubuntu by utilizing Docker and Docker Compose.

**Requirement**

Prior to starting, below are the subsequent measures to take

1. At Vultr, install a fresh Ubuntu 22.04 server.

2. In your DNS, construct a fully qualified domain name that points to the server's IP address.

3. Throughout the manual, you will find the URL joplins.examples.com mentioned repeatedly.

4. Access the server utilizing an SSH connection as a non-root user endowed with **sudo** privileges.

5. Set up Docker Compose in conjunction with Nginx.

6. Fundamental knowledge of Docker and managing systems.

**1. Joplin Server Installation**

* Initiate a fresh folder in the /opt directory to house the Joplin Docker files

*$ sudo mkdir /opt/Joplin*

* Go to the directory.

*$ cd /opt/joplin/*

* Employ a text editor to generate a fresh Joplin Docker Compose document

*$ sudo nano joplin-docker-compose.yml*

* Include the following information within the document
* Feel free to modify the numbers 5434 and 8080 as needed; these are merely illustrations
* Substitute https://joplins.examples.com with your complete domain name.
* Adjust the values of POSTGRES\_PASSWORD, POSTGRES\_DATABASE, and POSTGRES\_USER to your preferred configuration.

*version: '3'*

*services:*

*db:*

*image: postgres:13*

*volumes:*

*- ./data/postgres:/var/lib/postgresql/data*

*ports:*

*- "5432:5432"*

*restart: always*

*environment:*

*- POSTGRES\_PASSWORD=Your-Password-here*

*- POSTGRES\_USER=joplin-user*

*- POSTGRES\_DB=joplindb*

*app:*

*image: joplin/server:latest*

*container\_name: joplin-server*

*depends\_on:*

*- db*

*ports:*

*- "8080:8080"*

*restart: always*

*environment:*

*- APP\_PORT=8080*

*- APP\_BASE\_URL=https://joplins.examples.com*

*- DB\_CLIENT=pg*

*- POSTGRES\_PASSWORD=Your-Password-here*

*- POSTGRES\_DATABASE=joplindb*

*- POSTGRES\_USER=joplin-user*

*- POSTGRES\_PORT=5432*

*- POSTGRES\_HOST=db*

This Docker Compose setup establishes a fresh Postgres database container alongside a Joplin Server that's active on port 8080

* Save the document, then close it.
* Get the Joplin Server initiated

*$ sudo docker-compose -f joplin-docker-compose.yml up -d*

* Validate whether the Joplin Server is operational.

*$ sudo docker ps*

The expected outcome should align with the illustration presented below

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS | NAMES |
| 143a9e9cdaa5 | joplin/server:latest | "tini -- node dist/a…" | 2 minutes ago | Up 4 seconds | 0.0.0.0:8080->8080/tcp, :::8080->8080/tcp | joplin-server |
| 1e55ccad2d52 | postgres:13 | "docker-entrypoint.s…" | 2 minutes ago | Up 2 minutes | 0.0.0.0:5432->5432/tcp, :::5432->5432/tcp | joplin\_db\_1 |

**2. Configure a Reverse Proxy with Nginx**

* Establish a fresh configuration file for Nginx

*$ sudo nano /etc/nginx/conf.d/joplin-server.conf\*

* Incorporate the contents listed below into the file.

*server {*

*listen 80;*

*listen [::]:80;*

*server\_name joplins.examples.com.*

*error\_log /var/log/nginx/joplin-server.error.*

*location / {*

*proxy\_pass http://127.0.0.1:8080;*

*proxy\_set\_header Host $host;*

*proxy\_set\_header X-Real-IP $remote\_addr;*

*proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;*

*}*

*}*

* Save the document, and close it
* Examine the document's syntax and verify if there are any configuration problems with

Nginx.

*$ sudo nginx -t*

* To implement the adjustments, restart Nginx.

*$ sudo service nginx restart*

**3. Security**

UFW comes pre-activated on Ubuntu installations. Following are the procedures to adjust the firewall settings to permit HTTP and HTTPS connections on the server. Enable and permit access for both HTTP and HTTPS on the designated port let say 80 and 443 respectively. Followed by the refreshing the firewall rules to implement the modifications, which can be achieved by executing the below commands.

1. *$ sudo ufw allow 80/tcp*
2. *$ sudo ufw allow 443/tcp*
3. *$ sudo ufw reload*

**4. Enable HTTPS Access Configuration**

* Set up the Let's Encrypt Certbot client.

*$ sudo apt install python3-certbot python3-certbot-nginx -y*

* Get an SSL certificate for free. Change admins@examples.com to your email address and joplins.examples.com to your fully-qualified domain name.

*$ sudo certbot -d joplin.example.com -m admin@example.com*

* Certbot will initiate a request for a new certificate, install it on your Nginx server, and configure an automatic renewal process. For additional details, please refer to our Certbot guide
* Restart Nginx to implement the modifications.

*$ sudo service nginx restart*

**5. Access the Joplin Server**

* To assess the Joplin server, please open a web browser and go to your subdomain at: [**https://joplins.examples.com**](https://joplins.examples.com)
* Sign in to the Joplin server utilizing the default email and password credentials.

Email: admin@localhost

Password: admin

* Once you are logged in, click on your username, Admin, located in the upper right corner.
* You can then update your Administrator profile by filling in the Full name, Email, and Password fields. It is recommended to select a strong and secure password to enhance the Joplin server's security, followed by clicking Update Profile.
* Next, click on Admin in the top navigation menu.
* Head over to the Users tab and choose Add user to configure a new non-administrative account for synchronization with your Joplin Client
* Ensure the user form is filled out and then click on Create user to establish the new account.

**A screenshot of a computer

Description automatically generated**

**6. Link the Joplin Server and Clients**

Joplin provides clients that can be accessed on desktop as well as mobile platforms.

* Get the Joplin client application by visiting the official website.
* Initiate the Joplin client application.
* The Preferences menu is chosen by Mac users.
* The Settings menu is chosen by Windows users.
* The Configuration menu is chosen by mobile users.
* Go to synchronisation. then Click synchronisation target drop-down.

A screenshot of a computer

Description automatically generated

* Select the Joplin Server from the available options.
* In the Joplin Server URL section, type your fully qualified domain name.
* Enter the email address associated with your account in the Joplin Server email field.
* Input your password into the designated password field for the Joplin Server.
* Enter your preferred timing in the designated Synchronization interval section.
* Save the modifications made.
  + Users on desktop devices should click the **Apply**.
  + Mobile users, on the other hand, should tap **Check Synchronization Configuration**.

**7. Troubleshooting**

Using curl, you can check the host port of Joplin Server from the command line.

*$ curl 127.0.0.1:8080*

Below is a list of frequent problems you might encounter, accompanied by solutions.

**Bad Gateway**

One such issue you might encounter is a **Bad Gateway** error.

*502 Bad Gateway*

Verify that the Joplin Server container is operational and actively listening on the designated port specified in your Docker Compose file. For illustration, this article references port 8080. Ensure that the Joplin Server is functioning within Docker and is responsive on the host port.

**Invalid origin on 127.0.0.1**

You might encounter an error message that states:

*Invalid origin:* [*http://127.0.0.1:8080*](http://127.0.0.1:8080)

If this situation occurs, please investigate the Docker logs for any hidden issues that may be present.

*$ sudo docker logs joplin-server*

**Not a valid URL**

Furthermore, the Joplin client may display another error message, which is:

Error. Please check that URL, username, password, etc. are correct and that the sync target is accessible. The reported error was:

Not a valid URL: joplins.example.com/api/sessions

To address the above error, prepend https:// to the URL and double-check that your username and password are accurate.

**Invalid origin on fully-qualified domain name**

Another potential error you could face is:

*Invalid origin: https://joplins.examples.com*

Examine the - APP\_BASE\_URL= environment variable within the Joplin Docker Compose file to ensure that the appropriate domain name is specified for accessing the application. The variable should appear as illustrated below. Ensure you replace joplin1.example1.com with your fully-qualified domain name.

*- APP\_BASE\_URL=https://joplins.examples.com*

**Conclusion**

In summary, you have accomplished the hosting of the Joplin Server on an Ubuntu platform through Docker and have synchronized it with the Joplin Client application. You also have the capability to establish multiple user accounts and keep them synchronized with the server.