Affine Installation

- Install and Configure Affine

Install and Configure Affine

https://www.youtube.com/embed/iUTr-SbkBtU

> A word about Affine's language when it comes to self-hosting vs "cloud". It's a bit confusing, but when you set up a self hosted server, and then you see "Enable Cloud Sync" for a workspace, it really means your server, and does not mean the Affine servers. So when you set it up, you ware keeping everything local and self hosted.

Affine is a wonderfully powerful replacement for cloud offerings like Miro, Notion, and so many other costly systems out there today.

If you're not a self-hosting type person, but you're interested in finding a really great, privacy focus replacement for such offerings, then Affine is very likely a great fit for you and / or your organization. By using the online Affine offering, you're supporting open source, and the continued development of privacy respecting systems like Affine.

If, however, you are like me, you'll likely want to setup the open source (MIT LIcensed), self hosted version on your own hardware, or perhaps a cloud VPS like Digital Ocean. It's really quite straight forward to do, so let's look at what you'll need to get it done.

What You'll Need

- Hardware to run Affine on
- Docker and Docker Compose installed on the server
- (optional) A public IP address
- (optional) A domain or sub-domain name for your affine application
- (optional) A reverse proxy

If you're not sure what most of the above list means, not to worry, I'm wokring on a series that will help you get through the basics of self hosting. I'll be posting that all at once when it's ready, so you can binge it if you wish, to my channel. So make sure to **subscribe to the AwesomeOpenSource** YouTube channel to see it when it comes out.

Installation of Docker-CE and Docker Compose

You can easily install Docker-CE, Docker-Compose, Portainer-CE, and NGinX Proxy manager by using this quick install script I created and maintain on Github. Just use the command:

`wget https://gitlab.com/bmcgonag/docker_installs/-/raw/main/install_docker_nproxyman.sh`

To download the script to your desired host.

Change the permissions to make the script executable:

`chmod +x ./install_docker_nproxyman.sh`

and then run the script with the command:

`./install_docker_nproxyman.sh`

When run, the script will prompt you to select your host operating system, then will ask you which bits of software you want to install.

Simply enter 'y' for each thing you want to install.

At some point, you may be asked for your super user (sudo) password as well.

Allow the script to complete installation.

At this point, you might want to log out and back in, as this will allow you to use the `docker` and `docker-compose` commands without the need of sudo in front of them.

Installing Affine

First let's make a new folder for our application. I like to create a directory structure with a parent level "docker" folder that my application folders are all kept in. I do this for organization, but also because I can run multiple application on a single server using Docker.

`mkdir -p docker/affine`

The command above tells my system to check and see if there is already a directory called 'docker', and if so, use it. If not, then create that folder. If it finds an existing docker folder it will then check to see if a folder called 'affine' already exists, and if so, use it. If not, it will create that folder at the same time.

Now we can move into our affine folder with

`cd docker/affine`

We want to create a new file called 'compose.yaml`. This type of file is how we define the application we want to run with docker.

`nano compose.yaml`

Once you have the nano text editor open (or any text editor of your choice), we need to copy the below text into that file.

```
services:
 affine:
   image: ghcr.io/toeverything/affine-graphql:stable
   container_name: affine_selfhosted
   command:
    ['sh', '-c', 'node ./scripts/self-host-predeploy && node ./dist/index.js']
   ports:
    - '3010:3010' # <-- change the left side if your host already uses 3010
    - '5555:5555' # <-- change the left side if your host already uses 5555
   depends_on:
    redis:
      condition: service_healthy
    postgres:
      condition: service_healthy
   volumes:
    # custom configurations
    - ./.affine/self-host/config:/root/.affine/config
    # blob storage
    - ./.affine/self-host/storage:/root/.affine/storage
   logging:
    driver: 'json-file'
    options:
      max-size: '1000m'
   restart: unless-stopped
   environment:
    - NODE_OPTIONS="--import=./scripts/register.js"
    - AFFINE_CONFIG_PATH=/root/.affine/config
    - REDIS_SERVER_HOST=redis
    - DATABASE_URL=postgres://affine:<arandomsetofcharactersandnumb3rs>@postgres:5432/affine  # change
the password here, then copy it and paste it in the postgres section below for POSTGRES_PASSWORD
    - NODE_ENV=production
    - AFFINE_ADMIN_EMAIL=${AFFINE_ADMIN_EMAIL} # <-- you can fill in an email here, but it's optional
    - AFFINE_ADMIN_PASSWORD=${AFFINE_ADMIN_PASSWORD} # <-- you can fill in a password, but it's optional
    # Telemetry allows us to collect data on how you use the affine. This data will helps us improve the app and
provide better features.
    # Uncomment next line if you wish to quit telemetry.
    - TELEMETRY_ENABLE=false
 redis:
   image: redis
   container_name: affine_redis
```

```
    restart: unless-stopped
    volumes:
      - ./.affine/self-host/redis:/data
    healthcheck:
      test: ['CMD', 'redis-cli', '--raw', 'incr', 'ping']
      interval: 10s
      timeout: 5s
      retries: 5
  postgres:
    image: postgres
    container_name: affine_postgres
    restart: unless-stopped
    volumes:
      - ./.affine/self-host/postgres:/var/lib/postgresql/data
    healthcheck:
      test: ['CMD-SHELL', 'pg_isready -U affine']
      interval: 10s
      timeout: 5s
      retries: 5
    environment:
      POSTGRES_USER: affine
      POSTGRES_PASSWORD: <arandomsetofcharactersandnumb3rs> # paste the same password from above
here.
      POSTGRES_DB: affine
      PGDATA: /var/lib/postgresql/data/pgdata
```

Once you've made any necessary changes above you can save with CTRL + O, then press Enter to confirm, and exit with CTRL + X.

> NOTE: You need to change the postgres password in 2 places in the above file to a long, strong password.

Now, we'll do two commands. First, we can pull down the images from the repositories for the different application services needed (e.g. Affine, Redis, and Postgres).

`docker compose pull`

After that completes, we'll start our application running with

`docker compose up -d`

This command will bring up our containers, and run the services in the background for us.

You can monitor the logs of the containers as they start up by running

`docker compose logs -f`

I personally like to just run those last two commands in one line:

`docker compose up -d && docker compose logs -f`

This will start showing us the log output as soon as the containers are started. Either way, give the containers about a minute, maybe two, then go to the IP address of your host server on port 3010.

If you happened to change the left side port number from 3010 to a different port number, you'll want to go to that port instead.

In my case i went to **http://192.168.10.134:3010**

The first time you visit this URL you'll create a new user, and be taken immediately into the "admin" portion of the application. In here, I highly recommend you go to the Settings in the left hand menu, and on the main portion of the page untick the option to require email verification.

Once done, you can go back to the main affine page by removing everything after 3010 in the URL bar.

Once there, you'll see the Affine main application. Make sure to check out the User Interface overview in the video for a good introduction on how to use affine, and how to ensure it will sync data properly to your database.

Support My Channel and Content

Support my Channel and ongoing efforts through Patreon:

https://www.patreon.com/awesomeopensource

Buy me a Beer / Coffee:
https://paypal.me/BrianMcGonagill