Instalação e Configuração Appflowy docker

- Step-by-step Self Hosting Guides - From Zero to Production Appflowy
- Deploy AppFlowy Cloud

Step-by-step Self Hosting Guides - From Zero to Production Appflowy

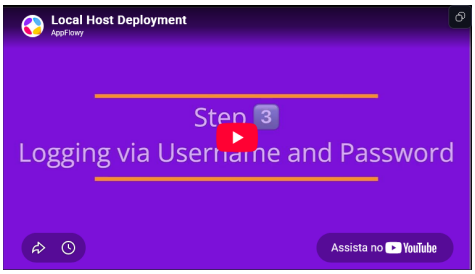
Link:https://appflowy.com/docs/Local-Host-Deployment

https://appflowy.com/docs/Step-by-step-Self-Hosting-Guide---From-Zero-to-Production
EM 23/08/2026

In this guide, we will deploy AppFlowy Cloud entirely on local machine. This deployment will not be accessible outside of the machine, and no collaboration is possible. However, it does allow a self hoster to quickly explore the editing functions offered by AppFlowy.

Local Host Deployment

https://youtu.be/V222xugM7oY?si=RIr-rPjkj73Ue0-Q



Prerequisite
Docker compose and Git are installed on your local machine.
Familiarity with command line.
AppFlowy desktop application is already installed.
Setting up the web services
Clone the AppFlowy Cloud repository.

```
git clone https://github.com/AppFlowy-IO/AppFlowy-Cloud.git
```

From the root of the directory, copy deploy.env to .env.

```
cp deploy.env .env
```

(Optional) If port 80 or 443 are already occupied by another service, such as Nginx, then modify the port binding via the following env variables.

```
NGINX_PORT=80
NGINX_TLS_PORT=443
```

Run the services via docker compose, in the background.

```
docker compose up -d
```

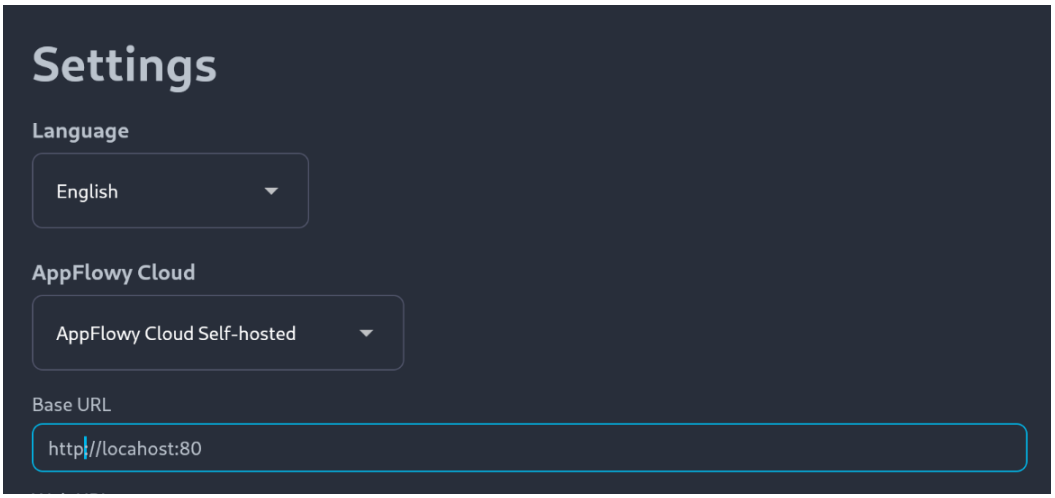
Check if all services are running and healthy.

```
docker compose ps
```

For subsequent sections, we will assume that port 80 is used for the deployment. If you have changed the port in earlier steps, please change the port accordingly.

Setting up the desktop client

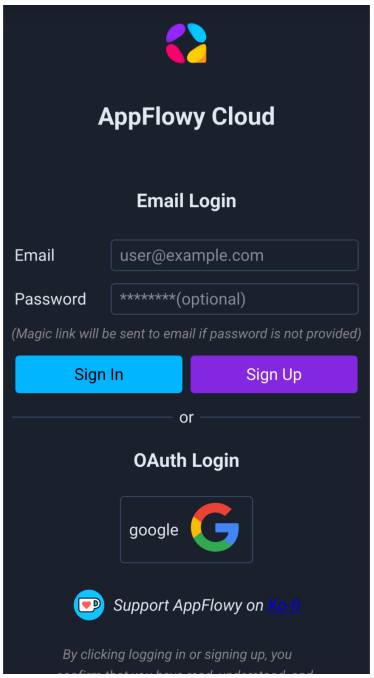
By default, the desktop application will connect to the official server. You can change the server endpoint via the settings, which can be found at the bottom of the login page.
Update the base URL to localhost:80. You may ignore the web URL for the time being.



There are multiple ways to log in to AppFlowy. We will start with password login, which work out of the box with the default configuration.
Close the application.

Log in via username and password

Currently, AppFlowy clients do not support password based login directly. However, it is still possible to login via username and password through the admin portal.
The admin portal is available on localhost:80/console. When setup correctly, you should land on a page similar to below:

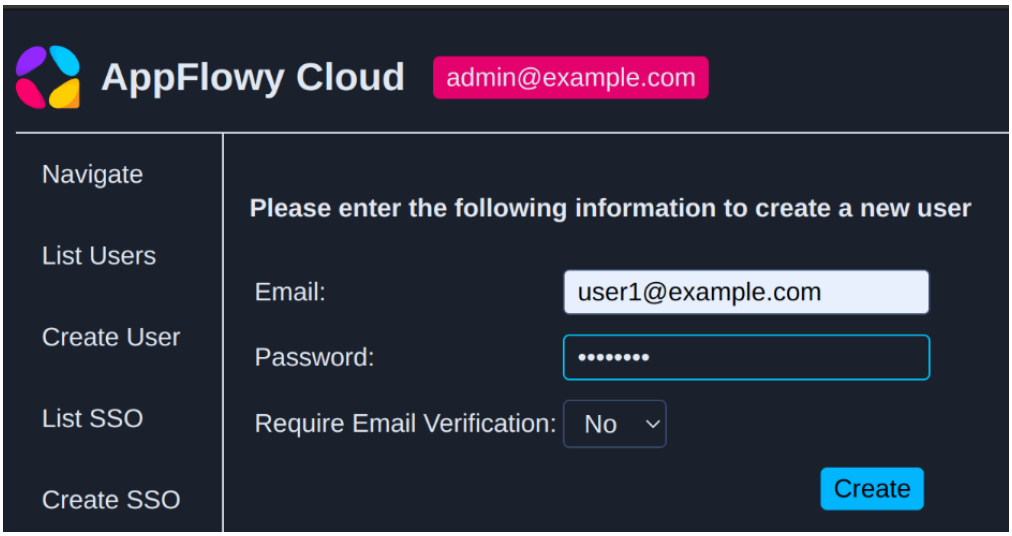


The credentials are set via the following variables:

```
GOTRUE_ADMIN_EMAIL=admin@example.com
GOTRUE_ADMIN_PASSWORD=password
```

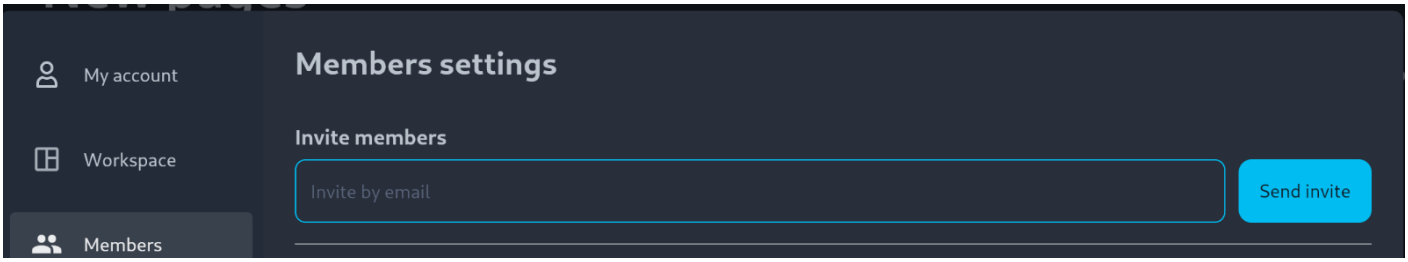
Do remember to change the default credentials if you plan to expose this deployment online. After changing the variables, the changes will take effect after you restart the services via docker compose.
Important: you should change the admin credentials **only** via the .env variable, and not via the admin console.
Navigate to the admin page. The admin page is accessible via the top right button in the home page:

Create a new user. The email doesn't need to actually exist, unless you plan to login using Identity F

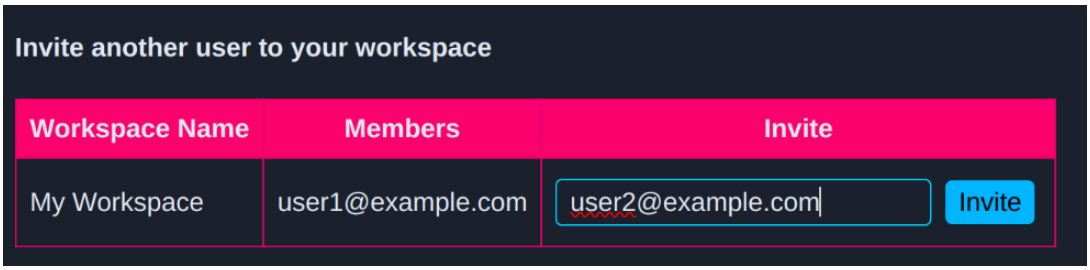


For the purpose of this demo, we will create two users, user1@example.com and user2@example.com .
Logout from the admin console, then log in using the credentials for user1@example.com.
Click on Open AppFlowy to launch the desktop application.

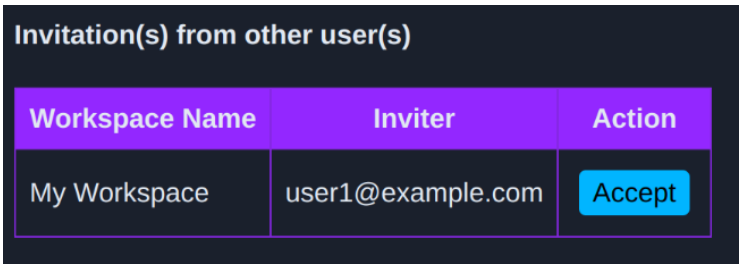
Inviting user to a workspace
If you have setup SMTP, you can invite a new member to your workspace directly from the workspace setting page, in the application:



Since we have not setup SMTP yet, for this guide, we will use the admin console to manage workspace member instead.
Login to the admin console using user1's credential.
On the left panel, click on Invite , then invite user2 to the workspace.



Logout from the desktop application, so that we can login as user2 later. Close the application.
Login to the admin console using user2's credential. You should see something similar to the image below, for the invite page:



Accept the invitation, then launch AppFlowy. user2 should now be a member of user1's workspace.

Shutting down service

The services can be stopped by running

```
docker compose down
```

Removing all data

If, at any point in time, you would like to erase all the data and start the setup from scratch again, run

```
docker compose down -v
```

➡ Next
$□□Setting up SMTP

Deploy AppFlowy Cloud

Link: https://github.com/AppFlowy-IO/AppFlowy-Cloud

https://appflowy.com/docs/self-host-appflowy-deploy-appflowy-cloud

Deployment
AppFlowy-Cloud is designed to be easily self deployed for self managed cloud storage
The following document will walk you through the steps to deploy your own AppFlowy-Cloud

Hardware Requirements

- Minimum 2GB Ram (4GB Recommended)
- Ports 80/443 available
- Because AppFlowy-Cloud will have to be running persistently (or at least whenever users need access),

  it's probably a good idea to run it on a non-end user device. It is best if you already have a home server (check software requirements),

  but if you don't, you can also deploy it on a cloud compute service as host server such as Amazon EC2 or Azure Virtual Machines

Software Requirements

Ensure you have Docker Compose(v2) installed on your host server. Follow the official guidelines for a reliable setup:
Docker Compose is included with Docker Engine:

Docker Engine: We suggest adhering to the instructions provided by Docker
for installing Docker Engine.
For older versions of Docker Engine that do not include Docker Compose:

Docker Compose: Install it as per the official documentation.
Once you have it installed, you can check by running the command:

```
docker compose version
# Docker Compose version 2.23.3
```

Note: docker-compose (with the hyphen) may not be supported. You are advised to use docker compose (without hyphen) instead.

Steps
□□□□□ Quick Try: Step-by-Step GuideFor an in-depth, step-by-step guide on self-hosting AppFlowy Cloud on AWS EC2, particularly for demonstration
purposes, please consult our detailed documentation:
Self-Hosting AppFlowy Cloud on AWS EC2 - Step by Step GuideNote: This guide is tailored for demonstration purposes using the AWS EC2 free tier. For more customized
deployment options, please follow the subsequent steps outlined below.

1. Getting source files

Clone this repository into your host server and cd into it

```
git clone https://github.com/AppFlowy-IO/AppFlowy-Cloud
cd AppFlowy-Cloud
```

2. Preparing the configuration

This is perhaps the most important part of the deployment process, please read carefully.
It is required that there is a .env file in the root directory of the repository.
To get started, copy the template deploy.env as .env using the following shell commands:

```
cp deploy.env .env
```

Kindly read through the comments for each option
Modify the values in .env according to your needs

For authentication details, refer to the Authentication documentation. You will need to update
the redirect URI to match your host server's public IP or hostname, such
as http://<your-host-server-public-ip-or-hostname>/gotrue/callback.

If using localhost, then just keep the default value.

```
GOTRUE_EXTERNAL_GOOGLE_ENABLED=true
GOTRUE_EXTERNAL_GOOGLE_CLIENT_ID=
GOTRUE_EXTERNAL_GOOGLE_SECRET=
GOTRUE_EXTERNAL_GOOGLE_REDIRECT_URI=http://your-
```

host/gotrue/callback

# GitHub OAuth2
GOTRUE_EXTERNAL_GITHUB_ENABLED=true
GOTRUE_EXTERNAL_GITHUB_CLIENT_ID=your-github-client-id
GOTRUE_EXTERNAL_GITHUB_SECRET=your-github-secret
GOTRUE_EXTERNAL_GITHUB_REDIRECT_URI=http://your-host/gotrue/callback

# Discord OAuth2
GOTRUE_EXTERNAL_DISCORD_ENABLED=true
GOTRUE_EXTERNAL_DISCORD_CLIENT_ID=your-discord-client-id
GOTRUE_EXTERNAL_DISCORD_SECRET=your-discord-secret
GOTRUE_EXTERNAL_DISCORD_REDIRECT_URI=http://your-host/gotrue/callback

3. Running the services

Start and run AppFlowy-Cloud
The following command will build and start AppFlowy-Cloud.

docker compose up -d

Please check that all the services are running

docker ps -a

4. Optional Services

We have provided optional services in the file docker-compose-extra.yml.

You do not need them for a fully functional installation of AppFlowy Cloud, but they could be helpful for various admin/debug tasks.

pgadmin (Web UI to visualize the provided postgres database)
portainer/portainer_init (Web UI to provide some monitoring and ease of container management)
tunnel (Cloudflare tunnel to provide a secure way to connect AppFlowy to Cloudflare without a publicly routable IP address)
admin_frontend (admin portal to manage accounts and add authentication methods. We recommend to keep this)

If you wish to deploy those, edit the file accordingly and do:

docker compose --file docker-compose-extras.yml up -d

You may ignore the orphan containers warning message that docker will output.
When using the docker compose up -d command without specifying a tag, Docker Compose will pull the latest tag for the appflowy_cloud and admin_frontend images from Docker Hub by default.

If you've set the APPFLOWY_CLOUD_VERSION or the APPFLOWY_ADMIN_FRONTEND_VERSION environment variable, it will pull the specified version instead.

Check that services are running correctly docker ps -a
If you find a particular service not working properly, you can inspect the logs:

Getting logs for a particular docker compose service

You can obtain its name by running `docker ps -a`
docker logs <NAME>
e.g. docker logs appflowy-cloud-admin_frontend-1

5. Reconfiguring and redeployment

It is very common to reconfigure and restart. To do so, simply edit the .env and run docker compose up -d again

6. Upgrading the services

To upgrade to the latest version, use docker compose pull and git pull to fetch the latest changes for images, docker compose files, and the configuration files.
Then, run docker compose up -d to start the services.

Alternatively, you can use a specific image tag instead of latest, and checkout the corresponding tag for the repository.

7. AppFlowy Web

AppFlowy Web is a Single Page Application (SPA) that calls the endpoints in appflowy_cloud, and is assumed to be served on a different origin that the one used for AppFlowy Cloud (eg. if you are hosting appflowy cloud on appflowy.home.com, appflowy_web may be hosted on web.appflowy.home.com). The source code and deployment guide can be found in this repository.

To prevent CORS issues, you will need to change set $appflowy_web "http://localhost:3000"; in nginx/nginx.conf to the origin where you are hosting appflowy_web.

Ports

After Deployment, you should see that AppFlowy-Cloud is serving 2 ports
443 (https)
80 (http)

Your host server need to expose either of these ports.

SSL Certificate

To use your own SSL certificates for https, replace certificate.crt and private_key.key with your own in nginx/ssl/ directory. Please note that the certificates in the repository are for demonstration purpose only and will need to be replaced by a certificate that is trusted by your devices.

For example, you can use Let's Encrypt, or CloudFlare Origin CA, if the AppFlowy Cloud endpoint is placed behind a cloudflare proxy.

Usage of AppFlowy Application with AppFlowy Cloud AppFlowy with AppFlowyCloud

FAQ

How do I use a different postgres?

The default url is using the postgres in docker compose, in service appflowy_cloud and gotrue respectively. However it is possible to use an external postgres, as long as it is accessible by the services. You need to change the following settings:

POSTGRES_HOST=postgres
POSTGRES_USER=postgres
POSTGRES_PASSWORD=password
POSTGRES_PORT=5432
POSTGRES_DB=postgres

You would need to run the supabase_auth.sh from migrations/before in your hosted postgres. Make sure that the environmental variables referenced by the script (eg. $SUPABASE_PASSWORD) have been

exported before running the script.

How do I disable signups?
If your deployed AppFlowy-Cloud is publicly available and you do not want any other users to access it, you can disable sign up by setting the GOTRUE_DISABLE_SIGNUP environment variable to true.

What port should I use for SMTP?
The default configuration assumes that TLS is used for SMTP, typically on port 465. If you are using STARTTLS, such as when using port 587, please change APPFLOWY_MAILER_SMTP_TLS_KIND to opportunistic.

Can I sign in using only using email and password?
The AppFlowy clients currently do not support email and password sign in. However, you can login to the admin portal using the admin email and password. In the admin section, you can then add users and set their passwords. Subseqently, users can login to the portal using their email and password, and launch the AppFlowy client via the portal.
What functionality will I lose if the SMTP server is not set up?
Sign in via magic link will not be possible. Inviting users to workspace and accepting invitation will have to be performed via the admin portal as opposed to links provided in emails.
I already have an Nginx server running on my host server. How do I configure it to work with AppFlowy-Cloud?
First, remove the nginx service from the docker-compose.yml file.
Update the docker compose file such that the ports for appflowy_cloud, gotrue, and admin_frontend are mapped to different ports on the host server. If possible, use firewall to make sure that these ports are not accessible from the internet.
Update proxy_pass in nginx/nginx.conf to point to the above ports.