

Resumes

- [Install Reactive Resumes](#)

Install Reactive Resumes

Installing Reactive Resumes

Setup a Server

In most self hosted applications these days, you'll need a machine to act as a server. This can be a machine in your home / business (such as an old laptop or desktop, a Raspberry PI or Single Board Computer (SBC), or even the computer you're reading this article from), a VM / Container hosted on one of your machines, or a VPS (Virtual Private Server) hosted by companies like RackNerd, Digital Ocean, Linode, Vultr, and so many more.

Regardless of which option you choose, you'll want to do a few things to get the server setup properly.

Install updates to our Server

Ubuntu / Debian

```
sudo apt update && sudo apt upgrade -y
```

RedHat / CentOS / Fedora / Alma / Rocky

```
sudo dnf update -y
```

Add a non-root / sudo user on the server

Generally, when you setup a new server, the VPS (Virtual Private Server) service sets up a default "root" user for you. It's considered unsafe to do everything as "root", so let's setup a non-root user who has super user (sudo) privileges.

```
adduser <username>
```

You'll be prompted to enter and confirm a password for this user. You'll also be asked for some user information like Name, etc, but this is not required information. At the end, confirm the entries, and you'll have your new user.

Next, we need to add the user to the super user group.

Ubuntu / Debian

```
usermod -aG sudo <username>
```

RedHat / CentOS / Fedora / Alma / Rocky

```
usermod -aG wheel <username>
```

Now, you can log out of the system, and log back in as your new non-root super user.

Install Docker and Docker Compose

Fortunately, there is a single line command that will install both Docker and Docker Compose for us on most Linux based distributions.

We need the 'curl' utility to get this to work, so if you don't have it, you'll want to install it first with

Ubuntu / Debian

```
sudo apt install curl -y
```

RedHat / CentOS / Fedora / Alma / Rocky

```
sudo dnf install curl -y
```

Next, we'll run the command to install Docker and Docker Compose:

```
curl https://get.docker.com | sh
```

You may be prompted to enter your super user password, so be ready for it. Once you do, the install should proceed.

Once complete, we want to add our user to the 'docker' group so we can do `docker` and `docker compose` commands without having to type in `sudo` each time.

```
sudo usermod -aG docker <username>
```

Now we'll logout / exit the session, and log right back in so the updated group will take effect.

Installing ReactiveResume

Let's create a folder structure in our server to help us get everything organized. From your home directory (you can use `cd` to go back to your home directory) we'll create a directory called 'docker', and within it a directory called 'resumes'.

```
mkdir -p docker/resumes
```

This command tells the system to create the 'docker' folder if one doesn't already exist, but if it does, then use it. Then it tells the system to create a folder named 'resumes' within the 'docker' folder if one doesn't already exist.

Now we'll move into our new folder with

```
cd docker/resumes
```

Finally, we'll create a new file called "compose.yaml" where we'll setup our application configurations.

```
nano compose.yaml
```

```
# In this Docker Compose example, it assumes that you maintain a reverse proxy externally (or
chose not to).
# The only two exposed ports here are from minio (:9000) and the app itself (:3000).
# If these ports are changed, ensure that the env vars passed to the app are also changed
accordingly.

services:
  # Database (Postgres)
  postgres:
    image: postgres:16-alpine
    restart: unless-stopped
    volumes:
      - ./postgres_data:/var/lib/postgresql/data
    environment:
      POSTGRES_DB: postgres
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: aL0n6Str0n6Pa5sw0rdtHa7isH4rdtoGu3s5
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U postgres -d postgres"]
      interval: 10s
      timeout: 5s
      retries: 5

  # Storage (for image uploads)
  minio:
    image: minio/minio:latest
    restart: unless-stopped
    command: server /data
    ports:
      - "9000:9000"
    volumes:
      - ./minio_data:/data
```

```

environment:
  MINIO_ROOT_USER: minioadmin
  MINIO_ROOT_PASSWORD: aDiff3r3n7L0n6Str0n6Pas5w0rd

# Chrome Browser (for printing and previews)
chrome:
  image: ghcr.io/browserless/chromium:v2.18.0 # Upgrading to newer versions causes issues
  restart: unless-stopped
  extra_hosts:
    - "host.docker.internal:host-gateway"
  environment:
    TIMEOUT: 10000
    CONCURRENT: 10
    TOKEN: chrome_token
    EXIT_ON_HEALTH_FAILURE: "true"
    PRE_REQUEST_HEALTH_CHECK: "true"

app:
  image: amruthpillai/reactive-resume:latest
  restart: unless-stopped
  ports:
    - "3000:3000"
  depends_on:
    - postgres
    - minio
    - chrome
  environment:
    # -- Environment Variables --
    PORT: 3000
    NODE_ENV: production

    # -- URLs --
    PUBLIC_URL: http://[your-server-ip]:3000 # alternatively the fqdn you want to access the
site with
    STORAGE_URL: http://[your-server-ip]:9000/default # alternatively the fqdn you want to
access the site with

    # -- Printer (Chrome) --
    CHROME_TOKEN: chrome_token
    CHROME_URL: ws://chrome:3000

```

```
# -- Database (Postgres) --
DATABASE_URL:
postgresql://postgres:aL0n6Str0n6Pa5sw0rdtHa7isH4rdtoGu3s5@postgres:5432/postgres

# -- Auth --
ACCESS_TOKEN_SECRET: [use the command 'openssl rand base64 32' to generate a token, then
copy and paste it here]
REFRESH_TOKEN_SECRET: [use the command 'openssl rand base64 32' to generate another
token, then copy and paste it here]

# -- Emails --
# if you want email to work, you need to uncomment the SMTP_URL line, and replace the
smtp url with the details of your email provider and account.
MAIL_FROM: noreply@localhost
# SMTP_URL: smtp://user:pass@smtp:587 # Optional

# -- Storage (Minio) --
STORAGE_ENDPOINT: minio
STORAGE_PORT: 9000
STORAGE_REGION: us-east-1 # Optional
STORAGE_BUCKET: default
STORAGE_ACCESS_KEY: minioadmin
STORAGE_SECRET_KEY: aDiff3r3n7L0n6Str0n6Pas5w0rd
STORAGE_USE_SSL: "false"
STORAGE_SKIP_BUCKET_CHECK: "false"

# -- Crowdin (Optional) --
# CROWDIN_PROJECT_ID:
# CROWDIN_PERSONAL_TOKEN:

# -- Email (Optional) --
DISABLE_SIGNUPS: "false" # change to true if you don't want other users signing up
(recommend you change it after you sign up your first user), and restart the container)
DISABLE_EMAIL_AUTH: "true" # change to false if you are setting up smtp email and want
email authentication

# -- GitHub (Optional) --
# GITHUB_CLIENT_ID: github_client_id
# GITHUB_CLIENT_SECRET: github_client_secret
```

```
# GITHUB_CALLBACK_URL: http://localhost:3000/api/auth/github/callback

# -- Google (Optional) --
# GOOGLE_CLIENT_ID: google_client_id
# GOOGLE_CLIENT_SECRET: google_client_secret
# GOOGLE_CALLBACK_URL: http://localhost:3000/api/auth/google/callback

# -- OpenID (Optional) --
# VITE_OPENID_NAME: OpenID
# OPENID_AUTHORIZATION_URL:
# OPENID_CALLBACK_URL: http://localhost:3000/api/auth/openid/callback
# OPENID_CLIENT_ID:
# OPENID_CLIENT_SECRET:
# OPENID_ISSUER:
# OPENID_SCOPE: openid profile email
# OPENID_TOKEN_URL:
# OPENID_USER_INFO_URL:
```

In the above file you really need to change some values before using this file.

1. Anything with a pound sign / hashtag in front of it is a "comment", which means it's ignored by docker.
2. Change the password values for Postgres and Minio. Note that these values are located in 2 places each, and the values for Postgres must match each other, and the values for Minio must match each other. The values for Postgres vs Minio should be different though.
3. You can use a command line command to generate the values for the ACCESS_TOKEN_SECRET and the REFRESH_TOKEN_SECRET. The command is `openssl rand base64 32`
4. If you want email in the system, you can change the MAIL_FROM address, and uncomment the SMTP_URL line, and replace the placeholder URL with your actual SMTP URL. The way it works is `smtp://<youremailuser>:<youremailpassword>@<yoursmtphosturl>:<yoursmtpportnumber>` .
5. If you want to disable signups for the site, you can change the value to 'true', but I suggest you do that after you have registered your first user, then restart the containers.

Once finished making your changes, use CTRL + O to save your changes, press Enter to confirm, and exit the nano editor with CTRL + X.

Pull the Docker Images

```
docker compose pull
```

Start the Containers and Follow the Logs

Now we'll start up our application, and follow the logs briefly, just to watch for any errors. We can do this by concatenating two commands into one line.

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

This will start your containers, and then show the log output as they start up. It will scroll by very fast, but we are just watching for anything we think may be an error.

If all has gone well, you should see a line that says the server has started, and is now accessible at an address and port.

This means we are able to go to our browser, and enter the IP address of our server, and the port number you specified in the port mapping. In my case I left the port at 3000, so I went to my server's IP with port 3000 like this:

```
http://192.168.10.84:3000
```

This brought me to the start page. You'll see the option to sign in, so click on it. Next, we need to create a new account. You should see a link to create a new account just above the username field. Click on it, then fill in the form with your name, username, email, and a password for the new account, then click 'Create'.

You can now start adding a new Resume.

Disable Sign Up

If you want to disable sign up now, then you can go back to the `compose.yaml` file, and you can change the value for `DISABLE_SIGNUPS` to be "true", then save your changes. Now restart the containers with

```
docker compose up -d --force-recreate
```

Don't worry, none of your data will be lost.