Baserow - No Code Database alternative to AirTable

https://www.youtube.com/embed/woTUh-6sEns

Baserow is a No-code, open source alternative to AirTable. If you've ever needed to collect data on anything. A system for sales leads, or a shopping list that does something specific to your needs, maybe a client support form, and a way to add comments, or perhaps a Product Management system where you can track, and be notified of changes to milestones, and goals...then Baserow may just be exactly the system you've been looking for.

This amazing little system gives you the ability to create a database, the tables that make up that data, and forms for users (or you) to utilize in filling in that data without writing a single line of code.

The system even provides you a simple link to share out with your users for the forms you create, or to allow them to see views of the data you are collecting.

What You'll Need

- Docker and Docker-Compose
- Email SMTP Setting Information
- (optional, but recommended) A domain / sub-domain name that you own.
- (optional, but recommended) NGinX Proxy Manager
- (optional) Portainer or Yacht for Docker Management in a GUI.
- About 20 minutes of your time.

Installation of Prerequisites

Installing Docker-CE and Docker-Compose

If you already have Docker and Docker-Compose installed, feel free to skip down to the next section.

You may want to install some pre-requisite softwre as well:

Debian / Ubuntu

```
sudo apt install git curl wget
```

Fedora / Redhat

```
dnf install git curl wget
```

Arch

```
sudo pacman -Sy git curl wget
```

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 permissios 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.

For instance, you may want to answer 'y' to NGinX Proxy Manager, and Portainer-CE if you don't already use these in your system.

At some point, you'll 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.

Installation of Baserow

Once you have docker and docker-compose installed and ready, we'll want to create a new folder for Baserow. I like to create a top level parent folder called docker in which I create all of my docker application folders. This allows me to keep them all together, and is easy to back up as a

single folder / zip file.

`mkdir -p docker/baserow`

Now move into the new baserow directory.

`cd ./docker/baserow`

Now we need to create three files.

1. Caddyfile - this is a small web server that will serve up the front end of our Baserow application in the browser.
2. .env - an environmental variable file. We use this file to set various configuratioin values for our install, such as the domain name, ports we want to use, and our email SMTP settings.
3. docker-compose.yml - the file we run to bring up our docker container(s) for our Baserow install.

The good news is that you only have to copy / paste for two of the files listed above. The .env file, however, you'll need to edit a few values in.

First, let's create the Caddyfile.

`nano Caddyfile`

Now, copy the information below by highlighting it with your mouse, then using CTRL + C to copy it, or right-click and select "copy" from the pop-up menu.

```
{
        {$BASEROW_CADDY_GLOBAL_CONF}
}

{$BASEROW_CADDY_ADDRESSES} {
        handle /api/* {
                reverse_proxy {$PRIVATE_BACKEND_URL:localhost:8000}
        }

        handle /ws/* {
                reverse_proxy {$PRIVATE_BACKEND_URL:localhost:8000}
        }

        respond /caddy-health-check 200

        handle_path /media/* {
```

```
                @downloads {
                        query dl=*
                }
                header @downloads Content-disposition "attachment;
filename={query.dl}"

                file_server {
                        root {$MEDIA_ROOT:/baserow/media/}
                }
        }

        reverse_proxy {$PRIVATE_WEB_FRONTEND_URL:localhost:3000}
}
```

Caddyfile

Now save the file with CTRL + O, then press Enter to confirm. Exit the nano editor with CTRL + X.

Next, we'll copy and paste the docker-compose.yml file.

`nano docker-compose.yml`

Again, copy the information from below, and paste it into the file you just created.

```
version: "3.4"

# == README ==
#
# To use this docker-compose.yml to run Baserow you must set the three required
# environment variables in the `x-backend-required-variables` section below and
review
# the variables in the `x-common-important-variables` section. If you receive the
# following error it is because you need to set the required environment variables
# first:
#   ```
# ERROR: Missing mandatory value for "environment" option interpolating
#   ```
#
#   If you are upgrading from Baserow 1.8.2 or earlier please read the additional
section
#   below.
```

```
#
#   See [Configuring Baserow](configuration.md) for information on the
#   other environment variables you can configure.
#
#   ## How to set environment variables
#
#   You can set these variables by using docker-compose env file
# (https://docs.docker.com/compose/environment-variables/#the-env-file):
#   1. Copy the `.env.example` file found in the root of Baserows repository
# (https://gitlab.com/bramw/baserow/-/blob/master/.env.example)  to `.env`:
#   ```
#   curl -o .env https://gitlab.com/bramw/baserow/-/blob/master/.env.example
#   ```
#   2. Edit `.env` and provide values for the missing environment variables.
#   3. `docker-compose up`
#
#   Alternatively you can set these variables by either running docker-compose with
# the environment variables set on the command line (fill in secure values first):
#   ```
#   SECRET_KEY= DATABASE_PASSWORD= REDIS_PASSWORD= docker-compose up
#   ```
#
# ## Upgrading from Baserow 1.8.2's docker-compose file
#
# To upgrade from 1.8.2's docker-compose file from inside the Baserow git repo you
need to:
#   1. Stop your existing Baserow install when safe to do so:
#   `docker-compose down`
#   2. `git pull`
#   3. Copy `.env.example` to `.env` and edit `.env` filling in the missing
variables
# below:
#   - `SECRET_KEY` to a secure value, existing logins sessions will be invalidated.
#   - `DATABASE_PASSWORD` to a secure password (this defaulted to 'baserow' before,
in
#     step 3 we are going to change the database users password to the value you
set)
#   - `REDIS_PASSWORD` to a secure password.
#   - `WEB_FRONTEND_PORT` back to 3000 if you want to continue accessing Baserow on
```

```
#     that port (it now defaults to 80).
#   - `BASEROW_PUBLIC_URL` to the URL/IP/Domain you were using access Baserow
remotely
#     (it must begin with http:// or https://). If you have set `WEB_FRONTEND_PORT`
to
#     anything but 80 you must append it to the end of `BASEROW_PUBLIC_URL`.
#   - `BASEROW_CADDY_ADDRESSES` configures which addresses the new internal Caddy
reverse
#     proxy listens on. By default, it will serve http only, enable automatic https
#     by setting to `https://YOUR_DOMAIN_NAME.com`. Append `,http://localhost` if
you
#     still want to be able to access Baserow from `localhost`.
#   4. Run the command below which will change the baserow postgresql users
password to
#      what you have set in step 1 in the .env file (no need to edit the command):
#   ```
#   docker-compose run --rm backend bash -c "PGPASSWORD=baserow psql -h db -U
baserow -c \"ALTER USER baserow WITH PASSWORD '$DATABASE_PASSWORD';\" && echo
'Successfully changed Baserow's db user password'"
#   ```
#   5. `docker-compose up -d`

# These environment variables are required to be set securely by you, see the
README
# above.
x-backend-required-variables: &backend-required-variables
  SECRET_KEY: ${SECRET_KEY:?
    !!!! ⚠IMPORTANT ⚠!!!! Open Baserow's docker-compose.yml and follow the
    instructions.

    This env var must be a condidential secret key used by Baserow's Django Backend
to
    secure signed data.}
  DATABASE_PASSWORD: ${DATABASE_PASSWORD:?Set to your own unique secure database
    password for the Baserow Postgresql service user to use, prior to version 1.9
    this defaulted to "baserow" and so should be set to that.}
  REDIS_PASSWORD: ${REDIS_PASSWORD:?Set to your own unqiue secure redis password
used
    to access the redis instance run by Baserow.}
```

```
# These are optional but important variables which control how Baserow can be
accessed.
x-common-important-variables: &common-important-variables
  # BASEROW_PUBLIC_URL is a comma separated list of urls you will be using to
access
  # Baserow, normally should just be a single URL. The first URL provided will be
used
  # to generate correct urls in emails, share links, downloads. All the urls will
be
  # used by the Caddy reverse proxy as addresses to listen for. See
  # https://caddyserver.com/docs/caddyfile/concepts#addresses for more info.
  # Caddy will setup HTTPS automatically if you have setup DNS correctly and use a
  # https:// url. See https://caddyserver.com/docs/automatic-https.
  # > If you are also setting WEB_FRONTEND_PORT to anything but 80 then you must
  # > append this port to your BASEROW_PUBLIC_URL like so:
  # > $YOUR_BASEROW_PUBLIC_URL:$YOUR_CUSTOM_WEB_FRONTEND_PORT
  BASEROW_PUBLIC_URL: ${BASEROW_PUBLIC_URL-http://localhost}
  # This is a comma separated list of addresses the Caddy reverse proxy sitting in-
front
  # of Baserow's services will listen on. If you wish to enable automatic https
then
  # you should change this to be `https://YOUR_DOMAIN_NAME.com`.
  BASEROW_CADDY_ADDRESSES: ${BASEROW_CADDY_ADDRESSES:-:80}
  # WEB_FRONTEND_PORT is the HTTP port that Baserow will bind to on the host to
serve
  # content. Prior to 1.9 this defaulted to 3000 but was changed to 80 in 1.9.
  # > If set to anything but the default of 80 to access Baserow you will need to
  # > append the port to your BASEROW_PUBLIC_URL (see above).
  WEB_FRONTEND_PORT: ${WEB_FRONTEND_PORT:-80}
  # WEB_FRONTEND_SSL_PORT is the SSL port Baserow will bind to on the host for
HTTPS.
  WEB_FRONTEND_SSL_PORT: ${WEB_FRONTEND_SSL_PORT:-443}
  # HOST_PUBLISH_IP is the address Baserow will bind to on the host, the choices
are:
  # - The default value of 0.0.0.0 : This will make Baserow accessible publicly
from
  #   other machines on your network, the internet if the server is exposed to it
and
```

```
  #   also due to a docker bug/"feature" bypass any Ubuntu UFW Firewall rules
  #   (See https://github.com/chaifeng/ufw-docker).
  # - 127.0.0.1 : This will make Baserow private and only available on the machine
  #               running docker-compose. Useful for when you want to run Baserow
behind
  #               a reverse proxy or just make it private.
  HOST_PUBLISH_IP: ${HOST_PUBLISH_IP:-0.0.0.0}
  # Set to any non-empty value to ensure Baserow generates https:// next links
provided
  # by paginated API endpoints. Baserow will still work correctly if not enabled,
this
  # is purely for giving the correct https url for clients of the API.
  # If you have setup Baserow to use Caddy's auto HTTPS or you have put Baserow
behind
  # a reverse proxy which:
  #  * Handles HTTPS
  #  * Strips the X-Forwarded-Proto header from all incoming requests.
  #  * Sets the X-Forwarded-Proto header and sends it to Baserow.
  # Then you can safely set BASEROW_ENABLE_SECURE_PROXY_SSL_HEADER=yes to ensure
Baserow
  # generates https links for pagination correctly.
  #
  # 1. !!! Do not enable this setting if you have not setup https. !!!
  # 2. See https://docs.djangoproject.com/en/3.2/ref/settings/#secure-proxy-ssl-
header
  #    for more details.
  # 3  You will also need to disable the automatic https provided by Caddy if using
your
  #    own reverse proxy by using the BASEROW_CADDY_GLOBAL_CONF variable above.
  BASEROW_ENABLE_SECURE_PROXY_SSL_HEADER:
${BASEROW_ENABLE_SECURE_PROXY_SSL_HEADER:-}

x-common-variables: &common-variables
  <<: *common-important-variables
  PRIVATE_BACKEND_URL: http://backend:8000
  PRIVATE_WEB_FRONTEND_URL: http://web-frontend:3000
  PUBLIC_BACKEND_URL:
  PUBLIC_WEB_FRONTEND_URL:
  FEATURE_FLAGS:
```

```
x-common-backend-variables: &common-backend-variables
  <<: *backend-required-variables
  <<: *common-variables
  MIGRATE_ON_STARTUP: ${MIGRATE_ON_STARTUP:-true}
  SYNC_TEMPLATES_ON_STARTUP: ${SYNC_TEMPLATES_ON_STARTUP:-true}
  DATABASE_USER: ${DATABASE_USER:-baserow}
  DATABASE_NAME: ${DATABASE_NAME:-baserow}
  ADDITIONAL_APPS:
  EMAIL_SMTP:
  EMAIL_SMTP_HOST:
  EMAIL_SMTP_PORT:
  EMAIL_SMTP_USE_TLS:
  EMAIL_SMTP_USER:
  EMAIL_SMTP_PASSWORD:
  FROM_EMAIL:
  DISABLE_ANONYMOUS_PUBLIC_VIEW_WS_CONNECTIONS:
  MEDIA_URL:
  BASEROW_EXTRA_ALLOWED_HOSTS:

services:
  # A caddy reverse proxy sitting in-front of all the services.
  caddy:
    image: caddy:2.4.6
    restart: unless-stopped
    environment:
      <<: *common-variables
      BASEROW_CADDY_GLOBAL_CONF: ${BASEROW_CADDY_GLOBAL_CONF:-}
    ports:
      - "${HOST_PUBLISH_IP:-0.0.0.0}:${WEB_FRONTEND_PORT:-80}:80"
      - "${HOST_PUBLISH_IP:-0.0.0.0}:${WEB_FRONTEND_SSL_PORT:-443}:443"
    volumes:
      - $PWD/Caddyfile:/etc/caddy/Caddyfile
      - media:/baserow/media
      - caddy_config:/config
      - caddy_data:/data
    healthcheck:
      test: [ "CMD", "wget", "--spider", "http://localhost/caddy-health-check" ]
      interval: 10s
```

```
      timeout: 5s
      retries: 5
    networks:
      local:

  backend:
    image: baserow/backend:1.10.0
    restart: unless-stopped
    environment:
      <<: *common-backend-variables
    depends_on:
      - db
      - redis
    volumes:
      - media:/baserow/media
    networks:
      local:

  web-frontend:
    image: baserow/web-frontend:1.10.0
    restart: unless-stopped
    environment:
      <<: *common-variables
    depends_on:
      - backend
    networks:
      local:

  celery:
    image: baserow/backend:1.10.0
    restart: unless-stopped
    environment:
      <<: *common-backend-variables
    command: celery-worker
    # The backend image's baked in healthcheck defaults to the django healthcheck
    # override it to the celery one here.
    healthcheck:
      test: [ "CMD-SHELL", "/baserow/backend/docker/docker-entrypoint.sh celery-
worker-healthcheck" ]
```

```
    depends_on:
      - backend
    volumes:
      - media:/baserow/media
    networks:
      local:

  celery-export-worker:
    image: baserow/backend:1.10.0
    restart: unless-stopped
    command: celery-exportworker
    # The backend image's baked in healthcheck defaults to the django healthcheck
    # override it to the celery one here.
    healthcheck:
      test: [ "CMD-SHELL", "/baserow/backend/docker/docker-entrypoint.sh celery-
exportworker-healthcheck" ]
    depends_on:
      - backend
    environment:
      <<: *common-backend-variables
    volumes:
      - media:/baserow/media
    networks:
      local:

  celery-beat-worker:
    image: baserow/backend:1.10.0
    restart: unless-stopped
    command: celery-beat
    # See https://github.com/sibson/redbeat/issues/129#issuecomment-1057478237
    stop_signal: SIGQUIT
    # We don't yet have a healthcheck for the beat worker, just assume it is
healthy.
    healthcheck:
      test: [ "CMD-SHELL", "exit 0" ]
    depends_on:
      - backend
    environment:
      <<: *common-backend-variables
```

```
    volumes:
      - media:/baserow/media
    networks:
      local:

  db:
    # Please ensure the postgres-client's major version in the backend image is
kept in
    # sync with this major version so pg_dump remains compatible.
    image: postgres:11.3
    restart: unless-stopped
    environment:
      - POSTGRES_USER=${DATABASE_USER:-baserow}
      - POSTGRES_PASSWORD=${DATABASE_PASSWORD:?}
      - POSTGRES_DB=${DATABASE_NAME:-baserow}
    healthcheck:
      test: [ "CMD-SHELL", "su postgres -c \"pg_isready -U ${DATABASE_USER:-
baserow}\"" ]
      interval: 10s
      timeout: 5s
      retries: 5
    networks:
      local:
    volumes:
      - pgdata:/var/lib/postgresql/data

  redis:
    image: redis:6.0
    command: redis-server --requirepass ${REDIS_PASSWORD:?}
    healthcheck:
      test: [ "CMD", "redis-cli", "ping" ]
    networks:
      local:

  # By default, the media volume will be owned by root on startup. Ensure it is
owned by
  # the same user that django is running as, so it can write user files.
  volume-permissions-fixer:
    image: bash:4.4
```

```
    command: chown 9999:9999 -R /baserow/media
    volumes:
      - media:/baserow/media
    networks:
      local:

volumes:
  pgdata:
  media:
  caddy_data:
  caddy_config:

networks:
  local:
    driver: bridge
```

docker-compose.yml

The file above is quite long. Please double check and make sure you got everything in it. There is nothing to be changed in this file, so save the file with CTRL + O, then press Enter to confirm. Exit the nano editor with CTRL + X.

Finally, let's create our .env file. Files that start with a period in the *nix based systems are hidden files, and therefore don't show up when you do a simple list command like `ls`, so if you want to see them, you have to use `ls -al`.

`nano .env`

Once again, copy the information below, and paste it into the file you just created.

```
SECRET_KEY={ mysupersecretpasswordforbaserow }
DATABASE_PASSWORD={ mypasswordforbaserow }
REDIS_PASSWORD={ 123456789124578 }
BASEROW_PUBLIC_URL={ http://baserow.mysuperdomain.com }
BASEROW_CADDY_ADDRESSES=:80
WEB_FRONTEND_PORT=80
HOST_PUBLISH_IP=0.0.0.0

EMAIL_SMTP={ yes or no }
EMAIL_SMTP_HOST={ smtp.mysuperdomain.com }
EMAIL_SMTP_PORT={ 24, or 465, o 587 }
EMAIL_SMTP_USE_TLS={ yes or no }
```

```
EMAIL_SMTP_USER={ me@mysuperdomain.com }
EMAIL_SMTP_PASSWORD={ my-super-secret-long-strong-email-password }
FROM_EMAIL={ me@mysuperdomain.com }
```

.env file

So, you need to either change or set almost everything in this file. I have surrounded the items you definitely need to either set, or change, with curly braces above { }.

SECRET_KEY needs to be a long, strong, comples string of upper and lower case letters and numbers.

DATABASE_PASSWORD needs to be a long, strong, comples string of upper and lower case letters and numbers.

REDIS_PASSWORD needs to be a long mixed set of numbers only. At least 10 characters long.

BASEROW_PUBLIC_URL is the domain or IP address that you want to reach baserow on through your browser. If you only intend to use baserow on your local network, you can use the IP of your host machine, otherwise you'll want to use a domain name that you own.

BASEROW_CADDY_ADDRESS - this needs to be 80 if possible, but if port 80 is already in use on your host, you'll want to change this port to an open / available port number.

WEB_FRONTEND_PORT - this needs to be left as 80, since Caddy calls it on 80.

HOST_PUBLISH_IP you can leave as 0.0.0.0, unless you have a reason to limit what machine IPs will be allowed to call this application.

All of the Email SMTP settings need to be set properly according to your email smtp service / provider.

Once, you have set or changed all of the values as needed, save the file with CTRL + O, then press Enter to confirm. Exit the nano editor with CTRL + X.

Running Our Application

We'll use a simple command to start the application running. I like to tack on a second command that also allows me to see the output being logged as the application starts up, so I can watch for any possible errors.

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

The above command will start the application installing and running - `docker-compose up -d`. The second part of the command will then show the logs as the application starts up after the images

are downloaded - `docker-compose logs -f`.

You can exit the logging at any time with the CTRL + C hotkey combination. The application will continue to run in the background.

The app takes a little bit to get started. I think on a first run I have to wait about 10 minutes for the app to get running. Be patient.

Seting up our Reverse Proxy so we can use a Domain Name

I have other videos on how to setup NGinX Proxy Manager to work inside a home network, but if you already have it running from my install script, you just need to forward ports 80 and 443 through your internet router / firewall to the host machine where NGinX Proxy Manager is running.

Next, make sure your domain name has an A-Record set that points to your public IP address. You can find out your public IP by going to https://ipchicken.com/.

Now, when you type in your domain name, you should, at the very least, see the Congratulations page from NGinX.

Next, we'll go into NGinX Proxy Manager, and add a new Proxy Host.

Enter the subdomain / domain for your baserow install, then press tab.

Next, enter the IP of your host machine in the IP Adddress field, and then enter the port you set for the Caddy item in the .env file. If you didn't change it this will be 80.

Now, tick the options for Block Common Exploits, and Websocket Support. Click Save, and test your new proxy entry by clicking the domain name in the list.

If Baserow isn't done starting up you may get a 'Server Error' screen. Be patient, it may still be starting up.

Test every few minutes, and once you see the login screen, you know it's up and running.

Now we want to make our site https (for secure). So click on the three dots to the right end of your proy host entry in NGinX Proxy Manager, and select 'Edit' from the drop down list.

Now go to the "SSL" tab, and click the drop down that says 'None'. Select 'Request a New Certificate', and then tick the box for 'Force SSL', and make sure your email is etnered in the Email field, then tick the box to accept the LetsEncrypt Terms of Service. Click Save, and wait.

If the pop-up window goes away with no error, then click the domain name in the list again. You should now be accessing your new Baserow install through an SSL Encrypted connection.

Supporting Baserow and Getting a Discount

It's super important to support Open Source projects, and Baserow was good enough to provide a **discount code** available to you all to get 10% off of your first paid transaction for Premium, Advanced, or Enterprise license plans.

That's right, if you pay for 1 year of Premium, Advanced or Enterprise you'll get 10% off for that first year.

If you only want to go monthly at first, not to worry, you still get 10% off that first month by using the discount code they provided.

This offer is **good through August 20, 2022**, so jump in, try out Baserow, and get a paid version for some great advanced features!

Dicount Code: AWESOME_OPEN_SOURCE_BASEROW

Support my Channel and Content

Support my Channel and ongoing efforts through Patreon:

https://www.patreon.com/bePatron?u=234177

Revision #1
Created 2022-09-30 16:18:59 UTC by Brian McGonagill
Updated 2022-09-30 16:20:23 UTC by Brian McGonagill