Baserow docker

Banco de dados Baserow

- Instalação e configurações Baserow
  - Instalação Baserow

Instalação e configurações Baserow

Procedimentos de instalação e configurações Baserow

Instalação Baserow

Link: https://baserow.io/docs/installation%2Finstall-with-docker-compose

Install with Docker compose

> Any questions, problems or suggestions with this guide? Ask a question in our community or contribute the change yourself at https://gitlab.com/baserow/baserow/-/tree/develop/docs .

Quickstart

A configuração a seguir é a maneira mais fácil de implantar Baserow com docker-compose e apenas usa a imagem tudo-em-um e um único contêiner. Se você usar essa configuração, então você em vez disso, deve consultar o guia Instalar com o Docker sobre as especificações de como trabalhar com esta imagem.

```
version: "3.4"
services:
  baserow:
    container_name: baserow
    image: baserow/baserow:1.33.4
    environment:
      BASEROW_PUBLIC_URL: 'http://localhost'
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - baserow_data:/baserow/data
volumes:
  baserow_data:
```

The rest of this guide will instead deal with the default `docker-compose.yml` found in the root of our git repository which runs each Baserow service as a separate container.

Installing requirements

If you haven't already installed docker and docker-compose on your computer you can do so by following the instructions on https://docs.docker.com/desktop/ and https://docs.docker.com/compose/install/.

> Docker-compose version 1.19.0 and Docker version 19.03 are the minimum versions required by our provided files.

Downloading the Baserow example docker-compose.yml

You can download the example Baserow `docker-compose.yml` by either directly downloading the file from https://gitlab.com/baserow/baserow/-/blob/master/docker-compose.yml and running:

```
curl -o docker-compose.yml https://gitlab.com/baserow/baserow/-/raw/master/docker-compose.yml
curl -o .env https://gitlab.com/baserow/baserow/-/raw/master/.env.example
curl -o Caddyfile https://gitlab.com/baserow/baserow/-/raw/master/Caddyfile
# Edit .env and set your own secure passwords for the 3 required variables at the top.
gedit .env
docker-compose up -d
```

or by directly cloning our git repo so you can get updates easier:

```
cd ~/baserow
git clone --depth=1 --branch master https://gitlab.com/baserow/baserow.git
cd baserow
cp .env.example .env
# Edit .env and set your own secure passwords for the 3 required variables at the top.
gedit .env
docker-compose up -d
```

```
# To update to the latest run:
docker-compose down
git pull
docker-compose up -d
```

> There is a security flaw with docker and the ufw firewall. By default docker when exposing ports on 0.0.0.0 will bypass any ufw firewall rules and expose the above container publicly from your machine on its network. If this is not intended then please set HOST_PUBLISH_IP to 127.0.0.1 so Baserow can only be accessed from the machine it is running on. Please see https://github.com/chaifeng/ufw-docker for more information and how to setup ufw to work securely with docker.

Usage

To use this docker-compose.yml to run Baserow you must set the three environment variables `SECRET_KEY`, `DATABASE_PASSWORD` and `REDIS_PASSWORD`. See the section below for more details. 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 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/baserow/baserow/-/blob/master/.env.example) to `.env`:

```
curl -o .env https://gitlab.com/baserow/baserow/-/raw/master/.env.example
```

2. Edit `.env` and provide values for the missing environment variables.

3. `docker-compose up`

Como alternativa, você pode definir essas variáveis executando docker-compose com o Variáveis de ambiente definidas na linha de comando (preencha os valores seguros primeiro):

```
SECRET_KEY= DATABASE_PASSWORD= REDIS_PASSWORD= docker-compose up
```

Upgrading from Baserow version 1.9.0 or later

1. It is recommended that you backup your data before upgrading, see the Backup sections below for more details on how to do this.
2. Stop your existing Baserow install when safe to do so: `docker-compose down`
3. Get the latest Baserow version by running: `git pull`
4. Startup the new version of Baserow by running: `docker-compose up -d`
5. Monitor the logs using: `docker-compose logs -f`
6. Once you see the following log line your Baserow upgraded and is now available again:

```
[BASEROW-WATCHER][2022-05-10 08:44:46] Baserow is now available at ...
```

Upgrading from Baserow 1.8.2's docker-compose file

> If you were previously using a separate api.your_baserow_server.com domain this is no longer needed. Baserow will now work on a single domain accessing the api at YOUR_DOMAIN.com/api.

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`

How To

Running management commands

You can see and run the Baserow backend management commands like so:

```
docker-compose exec backend /baserow/backend/docker/docker-entrypoint.sh help
```

View the logs

```
$ docker-compose logs
```

Run Baserow alongside existing services

Baserow's docker-compose files will automatically expose the `caddy` service on your network on ports 80 and 433 by default. If you already have applications or services using those ports the Baserow service which uses that port will crash. To fix this you can set the `WEB_FRONTEND_PORT` variable to change the default of port 80 and `WEB_FRONTEND_SSL_PORT` to change the default port of 443.

```
$ WEB_FRONTEND_SSL_PORT=444 WEB_FRONTEND_PORT=3000 docker-compose up
```

Using a Domain with automatic https

If you have a domain name and have correctly configured DNS then you can run the following command to make Baserow available at the domain with automatic https provided by Caddy.

> Append `,http://localhost` to BASEROW_CADDY_ADDRESSES if you still want to be able to access your server from the machine it is running on using http://localhost. See Caddy's Address Docs for all supported values for BASEROW_CADDY_ADDRESSES.

```
BASEROW_PUBLIC_URL=https://www.REPLACE_WITH_YOUR_DOMAIN.com \
BASEROW_CADDY_ADDRESSES=:443 \
docker-compose up
```

Behind a reverse proxy already handling ssl

```
WEB_FRONTEND_SSL_PORT= \
BASEROW_PUBLIC_URL=https://www.REPLACE_WITH_YOUR_DOMAIN.com \
docker-compose up
```

On a nonstandard HTTP port

```
WEB_FRONTEND_PORT=3000 \
BASEROW_PUBLIC_URL=https://www.REPLACE_WITH_YOUR_DOMAIN.com:3000 \
docker-compose up
```

Disable automatic migration

You can disable automatic migration by setting the `MIGRATE_ON_STARTUP` environment variable to `false` (or any value which is not `true`) like so:

```
MIGRATE_ON_STARTUP=false docker-compose up -d
```

Run a one off migration

```
# Use run if you have stopped your docker-compose environment
docker-compose run backend manage migrate
# Use exec otherwise
docker-compose exec backend /baserow/backend/docker/docker-entrypoint.sh manage migrate
```

Disable automatic template syncing

You can disable automatic baserow template syncing by setting the `BASEROW_TRIGGER_SYNC_TEMPLATES_AFTER_MIGRATION` environment variable to `false` (or any value which is not `true`) like so:

```
BASEROW_TRIGGER_SYNC_TEMPLATES_AFTER_MIGRATION=false docker-compose up -d
```

Back-up your Baserow DB

1. Please read the output of `docker-compose run backend manage backup_baserow --help`.
2. Please ensure you only back-up a Baserow database which is not actively being used by a running Baserow instance or any other process which is making changes to the database.

```
mkdir ~/baserow_backups
# The folder must be the same UID:GID as the user running inside the container, which
# for the local env is 9999:9999, for the dev env it is 1000:1000 or your own UID:GID
# when using ./dev.sh
sudo chown 9999:9999 ~/baserow_backups/
docker-compose run -v ~/baserow_backups:/baserow/backups backend backup -f
/baserow/backups/baserow_backup.tar.gz
# backups/ now contains your Baserow backup.
```

Restore your Baserow DB from a back-up

1. Please read the output of `docker-compose run backend manage restore_baserow --help`
2. Please ensure you never restore Baserow using a pooled connection but instead do the restoration via direct database connection.

3. Make a new, empty database to restore the back-up file into, please do not overwrite existing databases as this might cause database inconsistency errors.

```
docker-compose run -v ~/baserow_backups:/baserow/backups backend restore -f /baserow/backups
```