Picture / Image Sharing

Picsur! Alternative to Imgur

Install and Configure Picsur!

https://www.youtube.com/embed/mJXlAry2TuY

When you want to reference images for sites like Lemmy (or Reddit if you're still there), or maybe a blog, forum, or other place where images aren't allowed to be uploaded, image hosting sites like Imgur become incredibly useful and important. I was thinking about how I could do this in my own self-hosted infrastructure, and how to keep my images local. I did a quick search and came across several suggestions, but none of them really seemed to fit what I wanted. Luckily, I stumbled across Picsur! during my search, and it seemed to be exactly what I wanted. Easy drag and drop uploads, most common image formats supported, and links generated for reference in various types of applications (html, markdown, bb-code, and more).

I did run across an odd issue after adding my first image, and I noticed I would only see the text (alt text) of the image on the page, and not the image itself. I checked, double checked, and triple checked the formatting and spelling, and finally resorted to opening the developer tools on Firefox. Immediately I see a CORS error in the console.

What the heck is CORS?

No, it's not an American Beer, that's Coors. CORS stands for Cross Origin Resource Sharing. This is a special security precaution done by websites to keep other websites from pulling the resources and making calls improperly. It's really nice to have CORS in place, but it does cause a bit of an issue for us. Nothing we can't overcome, so stick with me.

CORS issues can be overcome programmatically, or by using the proper information in your web server, or in your reverse proxy. We'll be setting up our reverse proxy to help us overcome this issue today. I use NGinX Proxy Manager, so my instructions will be based on that application, but if you use other reverse proxies, the lines of information I provide will likely be the same, but setup in a different way for your chosen reverse proxy.

What You'll Need

- Docker and Docker Compose installed on the machine you want to use as your image server
- (Optional) NGinX Proxy Manager to use as a reverse proxy (or a reverse proxy of your choice)

- (Optional) a domain or sub-domain that you have access to set DNS A-records for
- About 20 minutes of your time

Installation of Docker, Docker Compose, and NGinX Proxy Manager via a Simple Script

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.

Install Picsur!

As always, I want to create my application folder for Picsur! under my parent level docker folder. In this case, since we'll be doing image storage, you may want to setup a different volume on a NAS or separate drive that has more space. If that's the case, not to worry, we just need to give that volume path in our docker-compose.yml file, and I'll point out where momentarily. First, let's create our folder structure.

`mkdir -p docker/picsur`

The command above says to check and see if the folder "docker" already exists, and if it does, use it, otherwise create it. Then the command moves to the next part of our folder path, and again checks to see if the folder "picusr" exists inside of the docker folder, and again tells the system if the folder exists, use it, otherwise create it.

Now we'll move into our new folder with

`cd docker/picsur`

Here we'll create our new docker-compose.yml file. I use the nano text editor directly in the terminal, but feel free to use any text editor you prefer.

`nano docker-compose.yml`

Copy the following block of yaml code into the file:

```yaml
version: '3'
services:
  picsur:
    image: ghcr.io/caramelfur/picsur:latest
    container_name: picsur
    ports:
      - '8080:8080' # change the left side of this mapping if 8080 is in use on the host system already
    environment:
      PICSUR_HOST: '0.0.0.0'
      PICSUR_PORT: 8080

      PICSUR_DB_HOST: picsur_postgres
      PICSUR_DB_PORT: 5432
      PICSUR_DB_USERNAME: picsur
      PICSUR_DB_PASSWORD: nice-long-strong-passw0rd-here  # must match the POSTGRES_PASSWORD in the
  picsur_postgres: section
      PICSUR_DB_DATABASE: picsur

      ## The default username is admin, this is not modifyable
      PICSUR_ADMIN_PASSWORD: different-nice-long-str0n6-passw0rd-here

      ## Optional, random secret will be generated if not set
      # PICSUR_JWT_SECRET: CHANGE_ME
      # PICSUR_JWT_EXPIRY: 7d
```

```
    ## Maximum accepted size for uploads in bytes
    PICSUR_MAX_FILE_SIZE: 128000000  # 128 MB
    ## No need to touch this, unless you use a custom frontend
    # PICSUR_STATIC_FRONTEND_ROOT: "/picsur/frontend/dist"


    ## Warning: Verbose mode might log sensitive data
    # PICSUR_VERBOSE: "true"
  restart: unless-stopped


  picsur_postgres:
    image: postgres:14-alpine
    container_name: picsur_postgres
    environment:
      POSTGRES_DB: picsur
      POSTGRES_PASSWORD: nice-long-strong-passw0rd-here
      POSTGRES_USER: picsur
    restart: unless-stopped
    volumes:
      # you can change the left side of the mapping below if you have images stored in a separate location
      - ./picsur-data:/var/lib/postgresql/data  # this will place all images in the same folder as docker compose
  volumes:
    picsur-data:  # if you change the volume mapping, make sure to change this to match
```

In the file above, you'll see a few places where I've added comments to help you make any changes you may need.

- Ports mapping from your host machine to the container
- password changes for the database
- password change for the admin user
- volume mapping change if you have some other location for your image storage.

Once you've made all the changes you need, save the file with CTRL + O, then press Enter to confirm, and exit the nano text editor with CTRL + X.

Now we are ready to start pulling down our Picsur! image, and running the container. We can do this with the command:

`docker compose up -d`

Once the image has pulled, and all of the container parts have downloaded, extracted, and started you'll be back at your command / terminal prompt. Give the application about 30 seconds to get up and running, then you can check that it's running by opening your favorite, modern, web browser, and navigating to the IP address of your host machine, and the port you entered on the

left side of the port mapping in your docker-compose.yml file. In my case I went to
http://192.168.10.60:8252 .

If all went well, you'll see a Picsur screen with a login option in the center of the view. You can click this login option and enter the username of `admin` and the password you entered in your docker-compose.yml file for the administrative user.

If you only intend to use the application locally, you can start uploading images, and getting the URL for those images to test in your various local applications. If, however, you want this to be available for use on the internet, you need to now create a reverse proxy entry so your desired domain or subdomain name will route you to your Picsur! install.

Setting up a Reverse Proxy

I'll be using NGinX Proxy Manager, an open source GUI built on top of NGinX meant to help in setting up Reverse proxies for all your services. If you've just installed it using my script, you'll want to login using those default credentials from teh script, and then change the credentials.

Next, head to the Proxy Hosts section, and click on 'Add Proxy Host'.

In the pop-up window, enter the domain name / subdomain you want for your Picsur site. I used ytimg.routemehome.org.

I own the domain routemehome.org. I have setup an A-Record that points my domain to the public IP address of my server. I route all requests that come in on ports 80 or 443 to the server running NGinX Proxy Manager. NGinX Proxy Manager then handles the requests and routes the requests to the proper servers / containers on my network.

In the "Forward Hostname / IP" field, enter the IP address of your server. If you've installed Picsur on the same server as NGinX Proxy Manager, you can enter "localhost" here.

Next, in the "Forward Port" field, enter the port number you have for the left side of the port mapping in your docker-compose.yml file. In my case, I used port 8252 instead of the default port 8080.

Now enable the options for "Block Common Exploits" and "Websockets Support".

Move to the "SSL" tab, and select "Request a New Certificate" from the dropdown that says "None". Next, enable the options for "Force SSL", "HTTP/2 Support", make sure your email is entered in the email field, and enable the option to accept the LetsEncrypt terms of service. Now click Save.

Wait while the Lets Encrypt system makes sure it can reach your application / service on Port 80, and issues you a CA Certified SSL certificate for your site. If all goes well, the pop-up will simply close with no other message. You should now be able to access your Picsur site at the domain name you entered. It should only come up through https, and should not give you any SSL certificate warnings.

One last step. For your Picsur entry, click the 3-dot icon at the right end of the row. Select 'Edit', and in the pop-up window navigate to the 'Custom Locations' tab.

On this tab enter "/" for the location, then enter the same IP address and Port number as you entered on the "Details" tab. Next, click on the little gear icon just above the "Forward Port" field. You'll see a new box open up, and in this box you need to add three lines. Feel free to simply copy them from here.

```
add_header Access-Control-Allow-Origin *;
proxy_hide_header cross-origin-resource-policy;
add_header cross-origin-resource-policy cross-origin;
```

This will allow any site to use image links from your Picsur install. If, however you only want a specific site to be able to use your image links, then change the asterisk "*" to the domain name or IP address of the site that you wish to allow access.

Again, click 'Save'.

Now you should be able to reach your Picsur site, and login to it. Upload a few images, then copy their links to use in a post or blog, and test out your new image server.

Support My Channel and Content

Support my Channel and ongoing efforts through Patreon: https://www.patreon.com/awesomeopensource