

Calendar and Availability Scheduling

- [Cal.com Availability Scheduling](#)
 - [Install and Setup Cal.com](#)
- [Easy Appointments Scheduling - Multiuser](#)

Cal.com Availability Scheduling

Install and Setup Cal.com

<https://www.youtube.com/embed/Niep6YkrkXA>

The ability to let others choose the best time to communicate with you is powerful. You start off the conversation in a meaningful way by giving your participant the ability to choose a date and time that best fits their schedule as well as dates and times that you've already set aside as best fitting your own.

Calendly is a closed source, proprietary application that provides this service, and Cal.com is its open source brethren. I was interviewing for new jobs a few years ago, and not sure what happened to just interviewing and either being hired or not, but at the time I went through all that. It was rounds of interviews with various people at the company. They used Calendly for scheduling the times to speak with them, and I found it quite compelling. I immediately start looking for an open source option, and pretty quickly found Calendso (now known as Cal.com).

I tried to get it running but ran into some issues, and upon looking at the Github Issues for the docker version of the project found I wasn't the only one. Here we are a couple of years later, and it's become much more stable, feature rich, and the docker based community version is fairly easy to get setup. You'll need a few things for this application to fully work for you, but it's a really amazing way to set aside time for meetings, and at the same time give some of the flexibility and power to those who want or need to meet with you.

What You'll Need

- A server or machine you want to host Cal.com on.
- Docker and Docker Compose installed on this server / machine.
- A reverse proxy (I use NGinX Proxy Manager)
- A domain name or subdomain for which you can create A / CNAME records for, on which you want to use for your Cal.com site.
- An SMTP server (your own, or one you can setup for SMTP email sending).
- About 30 minutes of your time

Installation

Installation of Docker and Docker Compose 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 -O install-docker.sh 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.sh
```

and then run the script with the command:

```
./install-docker.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. Enter the number associated with your host OS.

Next, simply enter 'y' for each thing you want to install. For this application, you should answer 'y' to "Docker-CE" and "Docker Compose" at a minimum. If you don't already have a reverse proxy running somewhere, you can also answer 'y' to "NGinX Proxy Manager".

At some point, you may be asked for your super user (sudo) password as well.

Allow the script to complete installation.

Once complete, 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.

Installing Cal.com

All instructions found in this section are available at <https://github.com/calcom/docker>. You should always check the source instructions for accuracy, particularly as this article ages.

You'll need "git" installed on this machine for the next part. You can install this as follows for the various Linux based distributions.

Ubuntu / Debian

```
sudo apt install git -y
```

Fedora / CentOS / RedHat

```
sudo dnf install git -y
```

OpenSuse

```
sudo zypper install git -y
```

Arch

```
sudo pacman -Sy install git
```

Next, we need to pull down the Cal.com docker git repository to our server. We can do this with the command:

```
git clone https://github.com/calcom/docker.git
```

Next, we want to change the folder name that was just created. "docker" is not a great name, so let's make it more descriptive.

```
mv docker cal-com-docker
```

We'll now move into our folder with

```
cd cal-com-docker
```

Now, we need to look at the files in this directory. We want to see them all, so we'll use the `-a` flag with the `ls` command. `-a` means show all.

```
ls -a
```

You should see several files and folders here. We are concerned with two files. The file named "docker-compose.yaml" and the file named ".env.example".

First we'll work with the environment variable file. We want to copy the example file, and then edit the copied file. This way if we mess something up, we'll still have the example file to start back at the beginning with.

```
cp .env.example .env
```

 <-- this command will copy the file ".env.example" to a new file called ".env".

Now, we'll edit the ".env" file.

```
nano .env
```

Here you'll see a large number of environment variables. These variables are used to setup the Cal.com application with the values we need in order for it to run in our selected environment. Carefully arrow through the comment, and set the values you need for each of the variables I define below.

NEXT_PUBLIC_LICENSE_CONSENT=true <-- this says you agree to the open source license terms.

LICENSE= <-- if you have purchased an enterprise level license, you would put your license key here, otherwise leave it blank.

NEXT_PUBLIC_WEBAPP_URL=https://yourcal.your-awesome-domain.com <-- you should remove this URL and put in the URL you want for your site. This needs to be a URL that can be reached by anyone who would be ideally able to setup a meeting with you.

NEXTAUTH_SECRET= <-- This needs to be created using the following command:

```
openssl rand -base64 32
```

This command will generate a key for you. Copy the output key to the file, and then double check that it's only 32 characters. If it's longer, just remove however many from the end of the key to make it only 32 characters. If you get it longer, or shorter, it will cause errors during setup.

CALENDSO_ENCRYPTION_KEY= <-- Again, this will be generated by a command, and again needs to be exactly 32 characters long.

```
dd if=/dev/urandom bs=1k count=1 | md5sum
```

Copy the output key to the file, and then double check that it's only 32 characters. If it's longer, just remove however many from the end of the key to make it only 32 characters.

POSTGRES_USER=unicorn_user <-- recommend changing this, it can be any username you want.

POSTGRES_PASSWORD=magincal_password <-- You should 100% change this to a long strong password with at least 16 characters or more.

POSTGRES_DB=calendso <-- you can leave this as is.

DATABASE_HOST=database:5432 <-- Unless you know what you are doing, leave this as is.

DATABASE_URL=postgresql://\${POSTGRES_USER}:\${POSTGRES_PASSWORD}@\${DATABASE_HOST}/\${POSTGRES_DB} <-- unless you know what you are doing, leave this as is.

GOOGLE_API_CREDENTIALS={} <-- unless you need to use this for some reason, leave this as is.

CALCOM_TELEMETRY_DISABLED=1

Set the following items, only if you are using the Microsoft Graph information for your calendar.

MS_GRAPH_CLIENT_ID=

MS_GRAPH_CLIENT_SECRET=

If you use ZOOM, then setup your Client ID and Secret here (you should create these for their API, **not your username and password**).

ZOOM_CLIENT_ID=

ZOOM_CLIENT_SECRET=

Set the from email that you want notifications to go out from here. If you are using GMail, Yahoo, Microsoft, etc, you'll likely have to use the email address you are actually sending from.

EMAIL_FROM=notifications@example.com

You need to setup the SMTP mail server settings here in order to send email. If you want to use GMail for this, you need to setup an application specific password for your gmail account. If you are using 2-factor authentication, that will likely cause issues. I am unable to help with those. I run my own mail server, and that's what I use to send emails.

EMAIL_SERVER_HOST=smtp.example.com < change this to the smtp server address for your mail provider.

EMAIL_SERVER_PORT=587 <-- make sure to set this port correctly, different servers will use ports like 25, 465, 587, and more. Check your provider's documentation for the correct settings.

EMAIL_SERVER_USER=email_user <-- This may be a username, or a full email depending on your email provider.

EMAIL_SERVER_PASSWORD=email_password <-- this would be the password for your email user.

NODE_ENV=production <-- leave this alone.

Save your changes with CTRL + O, then press Enter to confirm, and exit the nano editor with CTRL + X.

Whew! That's a lot of variables to get setup, but you really need to make sure the details are right for an application like this. Keep in mind, this application will sync scheduled appointments in the app with your calendar, as well as send emails to your users who are scheduling time, and also potentially setup online meeting spaces like Teams, Jitsi, Zoom, etc. So all of these pieces are necessary to get the application to run properly.

Next, let's make one small change in our docker-compose.yml file.

```
nano docker-compose.yml
```

In this file, we want to scroll down to the section under services >> calcom >> ports.

Here you'll see a port mapping of 3000:3000. This mapping equates to a port forward. The left side port is where your host machine (server) will be listening for connections for the Cal.com application. The right side is where the application listens for those connections in the docker container. So think of this as the server saying I got a request on port 3000, so I'm going to pass that along to the container on port 3000 as well.

What we want to do is change the host listening port (the left side) because 3000 is a very common port for node based applications to run on. So, let's change this port to something above 8080. In my case, I set it to 8594. Now my port mapping looks like

```
8594:3000
```

Save your changes with CTRL + O, then press Enter to confirm, and exit the nano editor with CTRL + X.

Setup our Reverse Proxy

Note: If you are running this on a VPS like [Digital Ocean](#), then you don't specifically need a reverse proxy, as you'll have a public IP address that you can point your A-record to in your domain registrar. Feel free to skip this section.

We are almost ready. We need to setup our reverse proxy so that it will point the domain name (subdomain) we set in the environment file to our application and port properly.

I'll be using NGinX Proxy Manager for this, so my instructions will be specific to it. If you are using a different reverse proxy, I expect you should know how to setup entries for your preferred software.

In NGinX Proxy Manager (NPM from here on), we'll want to click into 'Proxy Hosts', then click 'Add New Host' in the upper right.

In the modal (pop-up) window, enter the domain / subdomain name of your site. This needs to match what you entered in the .env file for NEXT_PUBLIC_WEBAPP_URL. Once entered press the Tab or Enter key so that the entry turns into a chip.

Now move to the IP Address field, and enter the private (LAN) IP address of your host machine (server).

Next, move to the port field, and enter the port you set on the left side of the port mapping in the docker-compose.yaml file. In my case I used 8594.

Now, enable the options for "Block common exploits", and "Websocket support".

Click 'Save'.

Start the Cal.com Application

It's time to start up our Cal.com web application. We do this with one command. Make sure you are in the cal-com-docker folder, and then enter the command:

```
docker compose up -d
```

Be patient, as this will pull down the images needed to create our containers, and depending on your internet speed and host machine capabilities, it will take several minutes. After that the application will start, and again, it may take a few minutes. If you don't receive any errors, then you should be good on startup.

After a few minutes, let's see if we can get to our Cal.com site by the URL we just setup in our reverse proxy.

If all went well, you should be greeted by an initial setup screen. We don't want to go through setup just yet though. First, let's get an SSL certificate so we are accessing our Cal.com site with strong encryption.

Go back into NPM and at the right end of the entry for our Cal.com url, click the vertical 3-dot icon, then select 'Edit' from the drop down menu.

In the modal (pop-up) select the SSL tab, and from the drop-down that says 'None', choose 'Request a new certificate'. Next enable the options for "Force SSL", "HTTP/2 Support", and both of the HSTS options. Make sure your email is entered, and enable the option to accept the LetsEncrypt TOS.

Click 'Save'.

This will take about 30 seconds to 1 minute, but the pop-up should close with no errors. You now have a LetsEncrypt certificate for your site. Refresh the page for your Cal.com site, and you should have the little Lock Icon.

Now, you'll need to go through the initial startup wizard. Follow my video to help you get through it if needed. Then there are a plethora of settings in the application you'll want to check out as well. During the Wizard if you get to the CalDav setup (and are using CalDav) and you hit an error, I show how to address it in the video as well.

Congratulations, you are now setup and ready to start accepting meeting with clients, prospects, interviewees, and more. Your imagination is the only limitation.

Support My Channel and Content

Support my Channel and ongoing efforts through Patreon:

<https://www.patreon.com/awesomeopensource>

Easy Appointments Scheduling - Multiuser

<https://www.youtube.com/embed/SfdGwj4ZHLQ>

Setup a Non Root User

```
adduser <username>
```

Enter your desired password for this new user, then confirm it.

Feel free to enter any of the other requested info, then accept the information entered.

Now make your user part of the 'sudo' group.

NOTE: If you are using a RedHat, CentOS, Fedora spin, then add them to the 'wheel' group instead.

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

Now you can log out from root, and log back in as the non-root user.

Install Docker and Docker Compose

Next we'll install Docker-CE and Docker Compose on the system. This can usually be done with a simple one-line command on most Linux distributions. Enter the following:

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

Enter your super user password if prompted, then allow the installation to complete.

Next, let's add your user to the 'docker' group.

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

Now, log out and back in, and you'll be able to run all the `docker` and `docker compose` commands without needing to use `sudo`.

Setup compose.yaml file

Now we need to create a simple folder structure, and then a file called "compose.yaml".

First, we'll make a folder for our docker container and data to be stored in.

```
mkdir -p docker/easyappt
```

This command creates the 'docker' folder if it doesn't already exist, then it will create the 'easyappt' folder inside of that 'docker' folder.

Now we'll move into the folder we just created:

```
cd docker/easyappt
```

Let's create our 'compose.yaml' file inside of it.

```
nano compose.yaml
```

Copy the code block below, and paste it into your new 'compose.yaml' file.

You'll need to make a few changes in the code block after you paste it. I have marked the placeholder items to be changed by surrounding them with less than '<' and greater than '>'.

It was pointed out that I didn't discuss how the application emails you, or the person scheduling your time. It appears it's pre-configured for php-mail. That said, you can setup and use your smtp email server if you have one, or have the information for one. (e.g. I use Purelymail, so I setup that one for my instance). I'll add a second compose.yaml file below to show the changes needed for email via smtp if you'd like to set that up).

```
---
services:
  easyappointments:
    image: alextseligidis/easyappointments
    environment:
      - BASE_URL=<http://sched.yourgreatdomain.org>
      - DEBUG_MODE=TRUE
      - DB_HOST=mysql
      - DB_NAME=easyappointments
      - DB_USERNAME=easyappt
      - DB_PASSWORD=<longstR0n6Pa5sw0rdTh4at1sUn1qu3>
    ports:
      - '8082:80' # you can change the left side port if needed.
```

```

# extra_hosts:
# - "<your.freeipa.server>:<your-server-local-ip"
mysql:
  image: mysql
  volumes:
    - ./data/mysql:/var/lib/mysql
  environment:
    - MYSQL_USER=easyappt
    - MYSQL_PASSWORD=<longstR0n6Pa5sw0rdTh4at1sUn1qu3>
    - MYSQL_ROOT_PASSWORD=<it-must-be-some-other-diffErent-paSsword>
    - MYSQL_DATABASE=easyappointments

```

In the above, make sure to change the password for the mysql user and root mysql user to something long and strong.

Also, if you need to use a port other than 80 on your server to reach the application, you can change the left side port number on the port mapping.

Finally, if you plan to use LDAP authentication, you may need to uncomment the lines for `extra_hosts:` and the line below it, filling in your ldap server hostname and domain and ip address appropriately.

Save the file with CTRL + O, then press Enter to confirm, and exit the nano text editor with CTRL + X.

For Email via SMTP in the YAML

```

---
services:

  easyappointments:
    image: alextselgidis/easyappointments:latest
    restart: unless-stopped
    ports:
      - '8082:80'
    environment:
      - BASE_URL=https://sched.yourgreatdomain.com
      - DEBUG_MODE=FALSE
      - DB_HOST=mysql
      - DB_NAME=easyappointments
      - DB_USERNAME=root
      - DB_PASSWORD=<a-sup3r-complex-lon6-pas5woRd>

```

```

- MAIL_PROTOCOL=smtp
- MAIL_SMTP_DEBUG=0
- MAIL_SMTP_AUTH=1
- MAIL_SMTP_HOST=<smtp.your-email-provider.org>
- MAIL_SMTP_USER=appointments@yourgreatdomain.org>
- MAIL_SMTP_PASS=<a-l0ng-str0n6-pa5sw0Rd-for-your-email>
- MAIL_SMTP_CRYPT0=tls # might need to be ssl
- MAIL_SMTP_PORT=587 # if ssl, use 465
- MAIL_FROM_ADDRESS=<appointments@yourgreatdomain.org>
- MAIL_FROM_NAME=Appointment
- MAIL_REPLY_TO_ADDRESS=<appointments@yourgreatdomain.org>
extra_hosts:
- "<ldap.your-ldap-domain.local>:<ip.for.ldap.server>"
mysql:
  image: mysql:8.0
  restart: unless-stopped
  environment:
    - MYSQL_ROOT_PASSWORD=<a-sup3r-complex-l0n6-pas5w0Rd>
    - MYSQL_DATABASE=easyappointments
  volumes:
    - ./mysql:/var/lib/mysql

```

In the above, you need to change the same information, surrounded by less than '<' and greater than '>' symbols. You also need to make sure to fill the proper details for your SMTP email provider. You can test the email is working by doing a 'Forgot Password' in the login screen. Enter the requested details, and it should successfully send you the new temporary password. This will be done, of course, after the initial setup screen has been completed.

Start our Docker Application

In the terminal enter the command:

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

The first part tells docker to bring up the containers in the background (detached or as a daemon), and the second part tells docker we want to follow the logs as they are generated while the application(s) start up.

The logs will likely scroll by very quickly, really you're just watching to see if you see any errors that may scroll past. We can always stop the logs and then scroll back up to read any errors if we see any. You shouldn't, but it does happen at times.

Once the logs slow down a bit, we can stop following them with CTRL + C.

Now let's go to our favorite modern browser, and open the IP and port we set for the application.

In my case I'll go to IP address 192.168.10.46, you'll likely have a different IP for your service. If you changed the left side of the port mapping in the compose.yaml file, then you'll want to put that after the IP in this form:

<http://192.168.10.46:8020> (for instance).

You should be greeted with the EasyAppointments first run wizard. We don't want to run through the wizard just yet, however. First we need to setup a domain or subdomain to point to our application so that our clients can access it from anywhere and schedule time with us.

Setup a DNS A-record and / or Reverse Proxy Entry

In order to ensure our site name resolves to our server and service we will need a DNS A-record (if you are running this on a server with a public IP address and are running it on port 80, then this will be all you need technically, but since we also want to give it a LetsEncrypt certificate, we'll use a reverse proxy as well) and a Reverse Proxy.

First, you need to own your domain, or at least have access to add DNS records for the domain you are setting up EasyAppointments for. Add an A-Record for the domain or subdomain and tell it to point to the public IP address where your server is running. If it's a VPS, such as on Linode, Digital Ocean, Hetzner, etc, then it likely has it's own public IP. If you're running inside of a local area network, then your ISP may issue a public IP address to your main modem / router, and this is the address you'll want to use.

The record will be entered slightly differently for each registrar, so if you want to call it sched.mygreatdomain.com, you need the following:

Type	Domain / Subdomain	IPv4 Address
A	sched	21.22.23.24

Of course, using your actual public IP address instead of 21.22.23.24.

Once that is set, it may take a few minutes to a few hours to take effect fully, so be patient. In the meantime we'll setup our Reverse Proxy entry. I use NGinX Proxy Manager, but feel free to use any reverse proxy you prefer.

With NGinX Proxy manager, I'll add a new entry for sched.mygreatdomain.com, then I'll enter my private IP address (the LAN address of the server where I'm running the EasyAppointments application, and the port (in my case 80, but remember to set the right port if you changed it in the compose.yaml file). Next, I'll tick the options for **Block Common Exploits**, and **Websocket Support**.

Finally, we'll move to the SSL tab. We'll change **None** to **Request a New Certificate**. Next tick the options for **ForceSSL**, **HTTP/2 Support**, and both **HSTS** options.

Accept the Terms of Service, enter your email. Before you click Save, you need to be sure your A Record is now functional and pointint to your external IP address. You also need to have forwarded port 80 and 443 through your firewall / router to the server / machine where you are running your reverse proxy.

Request for sched.mygreatdomain.com --> DNS A Record points to 21.22.23.24 --> Router Firewall points to Reverse Proxy at 192.168.1.25 inside your LAN --> Reverse Proxy sends request to 192.168.1.55:8020 inside your LAN and your user is presented with Easy Appointments.

If we have setup everything correctly, we'll be presented with the Easy Appointments First Run Wizard. Fill in the form as completely as you can. Click Next. Let the system setup all the database and new entries, and then you'll be ready to configure Easy Appointments.

I go through the setup in detail in the video, so be sure to check that out for initial configurations.

Plugins

EasyAppointments has several plugins in the 'Settings' area. You'll want to check those out. There may be one or more that you will find useful. I used the LDAP plugin to setup my FreeIPA system to allow Single Sign On for my Providers (what they call an employee essentially). I also setup calendar syncing with CalDav from NextCloud so as I add events they are synced to a specific NextCloud calendar, (but this will work with any CalDav), and when I add other appointments to my NextCloud calendar, they will sync to my EasyAppointments and block out that time.

Make sure to check out the video for more details on how to set this up. Be aware, when using OpenLDAP, or other LDAP systems, you should be familiar with your LDAP needs in order to properly configure your LDAP. If you are using FreeIPA, you can likely do the same as I did.

When setting up Caldav, you should also test first, then set it up for all your providers, or provide them detailed instructions on how to do it. It wasn't complicated, but testing first to make sure things work as expected is important.