Chat

Openfire XMPP Server

Openfire XMPP Server

Install and Setup Openfire XMPP Server

https://www.youtube.com/embed/pwIpDj9WwlI

Encrypted messaging has become a hot mess of one off platforms, and painful setups over the past 10 to 15 years. Before that, everyone, and I mean EVERYONE, used a standardized protocol that was awesome, and still is. XMPP was used by Microsoft, Yahoo, Excite, Google, Apple, and hundreds of other platforms for their chat communications. Then, for some reason they all started branching off, creating their own communication mechanisms and protocols, close-sourcing most of them, and segregating their users. We've had a few successes in bridging those chat gaps like WhatsApp (now owned by Facebook or Meta as they like to be called...so there went privacy), Telegram, Facebook Messenger (same story as WhatsApp), and on, and on. So, if you want to run a secure, encrypted server, you are really left to do it yourself. Every company that insists they'll stand by your privacy probably initially means it, but will eventually cave to the pressures of government, revenue, or who knows what else.

Today, we add to our collection of great chat options that are self-hostable, open source, and secure. Openfire is an XMPP server that provides a lot of control in the way you run it from the server side. It's not hard to get setup and running, and really does a terrific job of messaging.

What you'll need

- A machine or server to run Openfire on
- comfort with the CLI for a very simple install step.
- A MySQL/MariaDB Database (don't worry, we'll install and set it up in this article and the accompanying video)
- Either a local FQDN / Hostname for your server, or a publicly set FQDN / hostname for your VPS / Server with an A Record that points to it properly (public is optional)
- About 20 minutes of your time

Installation of Openfire

We will first download Openfire as a .deb (if you're using Arch, fedora, redhat, or other non-debian based distribution) simply look for the appropriate file in the downloads for OpenFire.

To pull this, we'll use wget to download it.

```
wget https://www.igniterealtime.org/downloadServlet?filename=openfire/openfire_4.7.4_all.deb -o openfire.deb
```

Once downloaded, we want to install it using the apt tool with the command:

```
sudo apt install ./openfire.deb -y
```

This will install the Openfire server, and start the service running inisitally. But, we want to make sure it runs each time the server is started, so let's set that up real quick with the command:

```
sudo systemctl enable --now openfire
```

Now, just to be safe, let's restart the service:

```
sudo systemctl restart openfire
```

and then check that's active with:

```
sudo systemctl status openfire
```

You should see the status as "active" here.

Installing MariaDB / MySQL

Next, we need to install a database for openfire to use. I'm going to install mariadb, but feel free to use mysql if you prefer to do so.

Luckily, MariaDB is in the debian/ ubuntu repositories already, so we can just run:

```
sudo apt install mariadb-server -y
```

Once installed, we need to secure the installation with the command:

```
mysql-secure-installation
```

You'll be prompted to enter the root password, but we haven't create a root password yet, so just press 'Enter' without typing anything in.

Next, you'll be asked if you want to convert to the 'unix socket' security model. Answer 'Y" for yes, and press enter.

Now enter a long, strong, complex password, then press enter. This is the password for your 'root' user. You'll be prompted to confirm the password as well, so make sure to store it in an encrypted password manager like Vaultwarden.

Now, answer each of the remaining questions with 'Y', then press enter. Once done, your MariaDB system is ready to use.

We need to do a few things in the MariaDB system to get it ready for Openfire. First, let's log into it. Use the command

```
mysql -u root -p
```

You'll be prompted for the root password you just created. Enter it, then press 'Enter'. You should now be logged into your MariaDB install.

You'll see the prompt with something like:

`| mariadb>` or `| mysql>`

Now we need to create the database and 'openfire' user, and a password for that user, as well as give that user privileges over the openfire database. We'll then run a script that was installed with the openfire server earlier, and it will do the rest of the work on the database for us.

1. Create the openfire database.

```
CREATE DATABASE openfire;
```

2. Give privileges to an new user called 'openfire', and assign a password (long and strong).

```
GRANT ALL PRIVILEGES ON openfire.* TO openfire@localhost IDENTIFIED BY 'longandstrongpasswordhere';
```

You really need to put this into a password manager or save it somewhere. You'll need it during the Openfire setup wizard.

3. Now let's clear and reset privileges on the database properly.

```
FLUSH PRIVILEGES;
```

4. Let's tell the system we want to use our new 'openfire' database

```
USE openfire;
```

5. Let's run the scripts

```
source /usr/share/openfire/resources/database/openfire_mysql.sql;
```

You should get a set of 'query ok' messages.

If so, you're ready to o to your new Openfire server in the web browser.

Run the Setup Wizard

Now, let's run through the Openfire setup wizard.

First, set your preferred language, then Continue. Next, You need to make sure that Openfire is setting the proper FQDN and XMPP Domain. You can also adjust the ports.

Next, ,select your preferred encryption, and give a strong encryption key.

Now, choose the Standard Database type. Leave the driver alone, but in the connection URL space, you'll want to change the "HOSTNAME" placeholder to the IP of your MySQL server, or use 127.0.0.1 if both servers are running on the same machine. Now, change the "DATABASE" placeholder to "openfire". Leave everything else alone. Enter the user as 'openfire', and the password you created for the 'openfire' user during the GRANT PRIVILEGES step above. Once entered, click 'Continue'.

Choose the type of profile settings you want, if using LDAP, you need to have an LDAP server, and now the settings to use it.

Now enter an email for your admin user, and a long, strong password for your admin user to log into the web admin system with. Click 'Continue / Finish', and you are done.

Now click the 'Login' button, and enter 'admin' as the user with the admin password you just created. You'll now be logged into the admin dashboard.

Check out the video for the overview, setting up users, and connecting clients to your server.

Support My Content on Patreon

Support my content and ongoing efforts on patreon. https://www.patreon.com/awesomeopensource

RocketChat

Self hosted Slack / Teams alternative

Install RocketChat

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

RocketChat is an open source, self hostable chat application that rivals the likes of Teams and Slack. It allows administratos incredible power in permissions and role setup, Room / Channel setup and organization, private / public channels, private group discussions, private 1 to 1 chats, audio / video integration with systems like Jitsi Meet, and so much more.

As a business, particularly in the tech service industry, your communication internally as well as with clients is key to success. A system like RocketChat can open communication avenues for you, your team, and your clients to have fast, live-time interactions.

What You'll Need

- A server to run Rocketchat On (ideally running Linux)
- Docker / Docker Compose installed
- A domain / sub-domain for your chat system to be reached on
- A Reverse Proxy
- About 30 minutes of your time.

Installation 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 RocketChat

First, we need to download two files. The docker compose file (compose.yaml) and the example .env file they provide. We can do that with the wget tool in our terminal.

Let's make a directory for our RoacketChat install and move into it

`mkdir -p docker/rocketchat`

`cd docker/rocketchat`

Now we can download the two files into our rocketchat directory.

```
wget https://raw.githubusercontent.com/RocketChat/Docker.Official.Image/master/compose.yml -O
compose.yaml

wget https://raw.githubusercontent.com/RocketChat/Docker.Official.Image/master/env.example -O
.env
```

Next, we need to set some of the environment variables to be specific to our needs for this RocketChat instance. We can edit the .env file using the nano editor in the terminal, or you can edit it with any text editor you prefer.

`nano .env`

Inside the .env file we need to uncomment a few lines. To do this, we remove the hastag (pound sign, number sign) at the beginning of the lines we need to edit. See my cleaned up example below and make yours look similar.

```
### Rocket.Chat configuration

# Rocket.Chat version
# see:- https://github.com/RocketChat/Rocket.Chat/releases
#RELEASE=
# MongoDB endpoint (include ?replicaSet= parameter)
#MONGO_URL=
# MongoDB endpoint to the local database
#MONGO_OPLOG_URL=
# IP to bind the process to
#BIND_IP=
# URL used to access your Rocket.Chat instance
ROOT_URL=https://chat.<your domain>.<org / com / net>
# Port Rocket.Chat runs on (in-container)
PORT=3000
# Port on the host to bind to
HOST_PORT=3000

### MongoDB configuration
# MongoDB version/image tag
MONGODB_VERSION=5.0
# See:- https://hub.docker.com/r/bitnami/mongodb

### Traefik config (if enabled)
# Traefik version/image tag
#TRAEFIK_RELEASE=
# Domain for https (change ROOT_URL & BIND_IP accordingly)
#DOMAIN=
# Email for certificate notifications
#LETSENCRYPT_EMAIL=
```

In the file above I have removed the hashtag in front of the lines for ROOT_URL, PORT, HOST_PORT, and MONGODB_VERSION. You should do the same. If you use the Traefik reverse proxy, uncomment and fill in those items as well. I use Nginx Proxy Manager, so those can remain commented out.

For ROOT_URL you should enter the url you want users to access the site from. For instance I set mine to "https://chat.sysmainit.com".

PORT and HOST_PORT can both be set as 3000 unless you already have something running on port 3000 on the system, in which case you should change HOST_PORT to some other port, and then make sure to use that port number in your reverse proxy setup.

Once you've made the necessary changes, you can save the file with CTRL + O, then press Enter to confirm, and finally close the nano editor with CTRL + X.

Next, let's quickly open the compose.yaml file with nano, and make a small modification there as well.

> NOTE: This change is only necessary if you won't be using the Traefik reverse proxy. If youdo use Traefik, then DO NOT make this change.

`nano compose.yaml`

In this file, we want to remove the section for the Traefik reverse proxy. If you are using Traefik as your reverse proxy, then feel free to leave it in.

```
volumes:
  mongodb_data: { driver: local }

services:
  rocketchat:
    image: registry.rocket.chat/rocketchat/rocket.chat:${RELEASE:-latest}
    restart: always
    labels:
      traefik.enable: "true"
      traefik.http.routers.rocketchat.rule: Host(`${DOMAIN:-}`)
      traefik.http.routers.rocketchat.tls: "true"
      traefik.http.routers.rocketchat.entrypoints: https
      traefik.http.routers.rocketchat.tls.certresolver: le
    environment:
      MONGO_URL: "${MONGO_URL:-\
        mongodb://${MONGODB_ADVERTISED_HOSTNAME:-
mongodb}:${MONGODB_INITIAL_PRIMARY_PORT_NUMBER:-27017}/\
        ${MONGODB_DATABASE:-rocketchat}?replicaSet=${MONGODB_REPLICA_SET_NAME:-rs0}}"
      MONGO_OPLOG_URL: "${MONGO_OPLOG_URL:\
        -mongodb://${MONGODB_ADVERTISED_HOSTNAME:-
mongodb}:${MONGODB_INITIAL_PRIMARY_PORT_NUMBER:-27017}/\
        local?replicaSet=${MONGODB_REPLICA_SET_NAME:-rs0}}"
      ROOT_URL: ${ROOT_URL:-http://localhost:${HOST_PORT:-3000}}
      PORT: ${PORT:-3000}
      DEPLOY_METHOD: docker
      DEPLOY_PLATFORM: ${DEPLOY_PLATFORM:-}
      REG_TOKEN: ${REG_TOKEN:-}
    depends_on:
      - mongodb
    expose:
      - ${PORT:-3000}
    ports:
      - "${BIND_IP:-0.0.0.0}:${HOST_PORT:-3000}:${PORT:-3000}"
  mongodb:
    image: docker.io/bitnami/mongodb:${MONGODB_VERSION:-5.0}
    restart: always
    volumes:
      - mongodb_data:/bitnami/mongodb
    environment:
      MONGODB_REPLICA_SET_MODE: primary
      MONGODB_REPLICA_SET_NAME: ${MONGODB_REPLICA_SET_NAME:-rs0}
      MONGODB_PORT_NUMBER: ${MONGODB_PORT_NUMBER:-27017}
      MONGODB_INITIAL_PRIMARY_HOST: ${MONGODB_INITIAL_PRIMARY_HOST:-mongodb}
      MONGODB_INITIAL_PRIMARY_PORT_NUMBER: ${MONGODB_INITIAL_PRIMARY_PORT_NUMBER:-27017}
      MONGODB_ADVERTISED_HOSTNAME: ${MONGODB_ADVERTISED_HOSTNAME:-mongodb}
      MONGODB_ENABLE_JOURNAL: ${MONGODB_ENABLE_JOURNAL:-true}
      ALLOW_EMPTY_PASSWORD: ${ALLOW_EMPTY_PASSWORD:-yes}
```

From the above we want to remove these lines:

```
labels:
      traefik.enable: "true"
      traefik.http.routers.rocketchat.rule: Host(`${DOMAIN:-}`)
      traefik.http.routers.rocketchat.tls: "true"
      traefik.http.routers.rocketchat.entrypoints: https
      traefik.http.routers.rocketchat.tls.certresolver: le
```

In nano, you can put your cursor on the line with "lables:", and then use the CTRL + K hotkey to remove the entire line. Just keeping pressing CTRL + K repeatedly until all of these lines are gone.

next, le'ts change the volume mapping line for our MongoDB database to be kept inside of our rocketchat folder. To do that change the line that says:

```
- mongodb_data:/bitnami/mongodb
```

to instead say

```
- ./mongodb_data:/bitnami/mongodb
```

This will keep our compose.yml, .env, and mongodb data all in this folder which makes it much easier to zip up and backup as needed.

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

Finally, we are ready to pull down the images for our RocketChat instance.

```
docker compose pull
```

As the images are pulled, be patient. It can take a bit of time as the images can be large sometimes, and then also have to be extracted (uncompressed).

Once the images are pulled down, we'll want to do one more thing. This is just based on my experience, but let's make that "mongodb_data" folder now, and give it the appropriate permissions it needs:

```
mkdir mongodb_data
```

```
chmod 777 ./mongodb_data
```

Let's start our system running with the command:

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

This is really two commands concatenated with the &&. First we tell docker to start the containers, and then we tell it we want to see the logs as it starts. We can stop seeing the logs using the CTRL + C hotkey combination.

Once it's up and running, you'll see a lot of text output. You are just looking for errors. If you don't see any errors, you can stop following the logs. Give the system a few minutes to be fully up and running.

Now, we just need to setup our reverse proxy.

We can open up NGinX Proxy Manager, and add a new Proxy Host. In this case I'm using the IP through my Netbird VPN, but you can use the local network IP, or the Public IP if you prefer. You just need to ensure that "chat.yourdomain.org" is pointing to your NGinx Proxy Manager instance. NGinX will handle the routing from there.

In our new proxy host, we'll name our entry with the domain name we want people to use to reach our site. In my case I entered "chat.sysmainit.com". You should use your domain.

Next we'll enter the IP we want NGinx proxy manager to use to reach the server where Rocketchat is running. After that enter the port that you set for the HOST_PORT variable. In my case I set 3000, so that's what I'll enter.

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

Next, select the SSL tab, and in the dropdown that says 'None', select 'Request a New Certificate'. Next, enable the options for 'Force SSL', 'HTTP/2 Support', 'HSTS for both of those entries, then make sure you email is filled in,and accept the LetsEncrypt terms of service.

Now click the 'Save' button. You should see the new proxy addition pop-up just close on its own after about 30 seconds. If you don't get any error in the pop up, your proxy is ready. You can now click on the entry in NGinX Proxy Manager, or simply go to the site in your favorite modern browser.

You'll be presented with a create account screen to create your first user. This user will be a system admin with all privileges needed to manage your server.

Check Out the video for an overview and walk through of the UI. There's a ton here, and you want to make sure to go through and get it setup right.

Support My Channel and Content

Support my Channel and ongoing efforts through Patreon:

https://www.patreon.com/awesomeopensource

Buy me a Beer / Coffee:
https://paypal.me/BrianMcGonagill