STEP BY STEP GUIDE FOR DEPLOY ROCKET.CHAT

(USING DOCKER AND DOCKER COMPOSE)

BY FELIPE ROSARIO DOS SANTOS

APPLYING FOR THE TECHNICAL SUPPORT SPECIALIST POSITION

São Paulo – Brazil

06/2023

This document was made to register all the steps to deploy a Rocket.Chat application using Docker & Docker Compose, integrate it with Github (Optional) use Nginx as a reverse proxy with SSL certificate, Ngrok to expose our localhost to the web, use Atlas MongoDB as the default database platform, and use Postman to test API calls.

All the Postman collections and API request results are available at:
https://github.com/FelipeRDEV/Rocket.Chat_ES_Felipe_Santos

Major software versions:

Linux Ubuntu 22.04 in a VirtualBox Machine connected via *bridge* to my host computer.
(I know that doesn't make much sense, since we're using Docker, but it's just for the test)
Docker - version 24.0.2
Docker Compose - version 2.11.2
MongoDB – version 6.0.6
Ngrok – version 3.3.1
Postman – version 10.15.0

PREPARING THE ENVIRONMENT

In this guide, we'll use the documents provided by Rocket.Chat to do the deployment: (https://docs.rocket.chat)

The first thing we need to do is install **docker** and **docker compose** in our machine, following the steps below:

1- Open a terminal and run this command to update all Ubuntu packages:

```
sudo apt update
```

2- Install pre-requirements to allow APT to use secure HTTPS packages

```
sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

3- Add the GPG key to make sure the packages that will come from docker repositories are valid:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

4- Add Docker repository to Ubuntu:

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

5- Update Ubuntu packages again

Sudo apt update

6- Install docker:

sudo apt install docker-ce

7- Verify the installation:

sudo systemctl status docker

docker –version

INSTALLING DOCKER COMPOSE

1- Create a directory to Docker Compose:

mkdir -p ~/.docker/cli-plugins/

2- Get the latest Docker Compose version from Github (In Rocket.chat webchat they recommend us to use 2.9.0 and above, in this case we're using 2.11.2):

curl -SL https://github.com/docker/compose/releases/download/v2.11.2/docker-compose-linux-x86_64 -o ~/.docker/cli-plugins/docker-compose

3- Give Docker Compose the necessary permissions:

chmod +x ~/.docker/cli-plugins/docker-compose

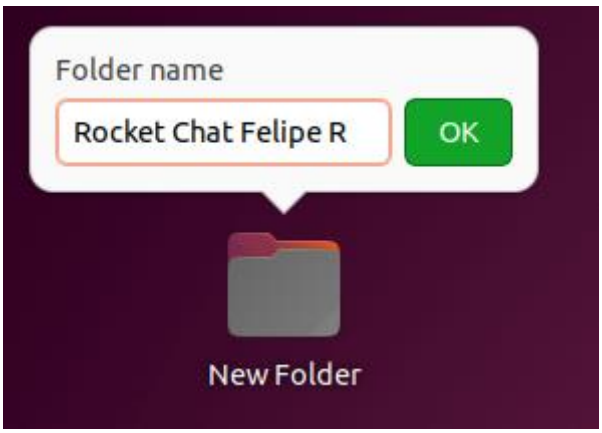
4- Verify the instalation:

docker compose version

After we installed Docker and Docker Compose, we need to fetch Rocket.Chat example compose file, by navigating to a directory of your choice and running this command in the terminal in the same folder:

curl -L https://raw.githubusercontent.com/RocketChat/Docker.Official.Image/master/compose.yml -O

By doing this, the compose.yml file will be created inside the folder:



(In my experience, when I cloned this file using curl with the url that is on the guide website, the file was slightly different from the example, I believe the version there is outdated, but it's easy to figure out how to edit it, although many fields in the example of the site did not appear in this most recent compose file).

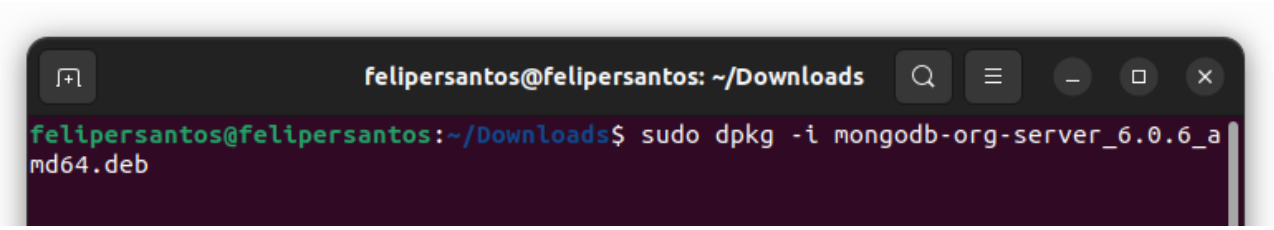
For now, we're not editing this file, since were a localhost Ip and local dependencies to host our application.

MONGODB INSTALLATION

Now we need to install MongoDb in our machine, to do that, we navigate to the following link and select the most compatible version with our OS:

https://www.mongodb.com/try/download/community

After downloading, we need to open a terminal instance in the same folder the download is, and run the dpkg command to “depackage” and install the files:



After installing, we can check MongoDB version, by using the command “mongod –version”:

SETUP THE HOST, INSTALL NGINX AND USE AN SSL CERTIFICATE

We'll use Nginx to do our reverse proxy and use a self-signed SSL certificate in a localhost.

1- Install UFW (Uncomplicated Firewall) if it's not installed by default:

sudo apt-get install ufw

IMPORTANT: We're going to add a firewall rule to permit your default SSH connection port on port 22/tcp.
In case you have the port changed on your device, be sure to use the corresponding port. Failure to do so will break your SSH connection and log you out of the server as soon as you enable the firewall!

2- Set the default access rules:

sudo ufw default deny incoming

sudo ufw default allow outgoing

3- Set the service rules (SSH / HTTPS):

sudo ufw allow 22/tcp

sudo ufw allow 443/tcp

4- Enable the firewall:

sudo ufw enable

5- Check the Firewall status:

sudo ufw status

```
felipersantos@felipersantos:~/Downloads$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW       Anywhere
443/tcp                    ALLOW       Anywhere
22/tcp (v6)                ALLOW       Anywhere (v6)
443/tcp (v6)               ALLOW       Anywhere (v6)
```

6- As we're making changes to the Firewall, we need to reload it:

sudo ufw reload

7- Install Fail2ban

(Fail2ban is an intrusion prevention software framework that protects computer servers from brute-force attacks).

sudo apt-get update

sudo apt-get install fail2ban

4. Editing the hosts file:

sudo nano /etc/hosts

If your hostname.domain.tld is mapped to a publicly routable IP, it needs to be set to your local address, for example, 127.0.0.1. Please note that the order in which localhost and your hostname are placed in this file is important; make sure localhost is first.

127.0.0.1 localhost localhost

127.0.1.1 felipersantos.server felipersantos

```
  GNU nano 6.2                    /etc/hosts *
127.0.0.1       localhost
127.0.1.1       felipersantos.server    felipersantos
```

INSTALLING NGINX & SSL CERTIFICATE

Install Nginx using this command:

sudo apt-get install nginx

We're going to use a self-signed SSL certificate, cause it's a quick way to have a secure HTTPS connection, for this purpose, we'll proceed using openSSL:

Open a terminal and execute this command:

openssl genrsa -des3 -out feliperosario.key 2048

*(You can name the .key file to whatever you want)***.**

Set a PEM password and continue.

If we do a "ls" in the terminal, we'll se that the .key file was created, now we have to make a csr file based on our key file:

openssl req -new -key feliperosario.key -out feliperosario.csr

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:BR
State or Province Name (full name) [Some-State]:Sao Paulo
Locality Name (eg, city) []:Embu
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Rocket Chat Felipe
Organizational Unit Name (eg, section) []:Rocket Chat Felipe
Common Name (e.g. server FQDN or YOUR name) []:felipersantos.server
Email Address []:feliperst.contato@gmail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:feliperkk2
An optional company name []:RocketChat1
```

After filling all the fields, we can see that the .**csr** file was created:

```
root@felipersantos:/home/felipersantos/Downloads# ls
feliperosario.csr  feliperosario.key  mongodb-org-server_6.0.6_amd64.deb
```

Every time Nginx starts its service, it will ask us for the PEM password, and that can be an issue, so we need to make a copy of the .key file and overwrite it with a new one that will not ask us for the password on every run:

cp feliperosario.key feliperosario.key.pw

openssl rsa -in feliperosario.key.pw -out feliperosario.key

```
root@felipersantos:/home/felipersantos/Downloads# cat feliperosario.csr
-----BEGIN CERTIFICATE REQUEST-----
MIIDMjCCAhoCAQAwgbUxCzAJBgNVBAYTAkJSMRIwEAYDVQQIDAlTYW8gUGF1bG8x
DTALBgNVBAcMBEVtYnUxGzAZBgNVBAoMElJvY2tldCBDaGF0IEZlbGlwZTEbMBkG
A1UECwwSUm9ja2V0IENoYXQgRmVsaXBlMR0wGwYDVQQDDBRmZWxpcGVyc2FudG9z
LnNlcnZlcjEqMCgGCSqGSIb3DQEJARYbZmVsaXBlcnN0LmNvbnRhdG9AZ21haWwu
Y29tMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAs83cmI4POrx2y9Cu
AE/PJ6oxsPu6bF4B60L+hTMOq3azluwMlKwuGqxeI2ECdPsMFUaua/Mh2/4FKBT7
oQsKp2p/8jqRQenKaTm2qUq0sSnSOtGBp8CJ6/i4uvtcS+tPgOJP87+/8fnCrpdj
PWoxqwZ2pivvQ0AdBj5RzbUlR3aVaKIqoug7oPaYdPnEHfNRxK15JEyv23mz6WYL
irrvmIzq7CPIaElW0+tjCMhbm+4US3wxoSyGIp7xrDqR/MN2z/+YF1j1kPmB0K8j
6YJMdvCRIAGKhEH9B1KNqkKkPqVokodyT4vK4GWP4P8BAGcvw3XgQx3fvAsah8oC
u3oQjwIDAQABoDcwGQYJKoZIhvcNAQkHMQwMCmZlbGlwZXJrazIwGgYJKoZIhvcN
AQkCMQ0MC1JvY2tldENoYXQxMA0GCSqGSIb3DQEBCwUAA4IBAQB7+yTaKD0B1kf+
Rzn0+D1JZ5I/o4Al5loJThVGqCIFGPgLM4vp2lctT6ERTdNEK1sYVmzM5A3rCcjZ
FZI2KoFOIUOP0wwz/DYx7Waji53robVPIiuUYkU98dCk6XiFUuRVibdqxnEZ3qIf
yc8IFlJzdiK7JGW4BBGKaRf/OktppRHsUiuj49wlOjWTPDWgjU4iaofXMCVf0QcM
/XImfmsqRGZZq/CaWCsG4WxViAZ9hRuyiVY+mH0R2BWbjrAXUFFWVd6dpJFiqZim
kuToI/b7c6xLUnRGdWZIDNTMSo3dcKb0OU2K+zzIzBTCuILgudhpeEirv/frjccf
d+XKjdoY
-----END CERTIFICATE REQUEST-----
```

We can visualize the .key file using "cat" command on Ubuntu.

Now we need to self-sign the using the x509 certificate.

openssl x509 -req -in feliperosario.csr -signkey feliperosario.key -out feliperosario.crt

```
root@felipersantos:/home/felipersantos/Downloads# openssl x509 -req -in feliperosario.csr -signkey feliperosario.key -out feliperosario.crt
Certificate request self-signature ok
subject=C = BR, ST = Sao Paulo, L = Embu, O = Rocket Chat Felipe, OU = Rocket Chat Felipe, CN = felipersantos.server, emailAddress = feliperst.contato@gmail.com
```

After signing, create a directory inside nginx folder, to copy the certificate files:

sudo mkdir /etc/nginx/ssl

sudo cp feliperosario.crt /etc/nginx/ssl

sudo cp feliperosario.key /etc/nginx/ssl

Now, we need to edit the default file inside the "sites-available" folder, and paste the code provided in Rocket.chat docs website:

sudo nano /etc/nginx/sites-available/default

Edit server_name, proxy_pass and SSL certificates path:

```
  GNU nano 6.2                                                    /etc/ngi
# HTTPS Server
    server {
        listen 443 ssl;
        server_name felipersantos.server;

        error_log /var/log/nginx/rocketchat_error.log;

        ssl_certificate /etc/nginx/ssl/feliperosario.crt;
        ssl_certificate_key /etc/nginx/ssl/feliperosario.key;
        ssl_dhparam /etc/nginx/dhparams.pem;
        ssl_protocols TLSv1.2;
        ssl_ciphers 'ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:
        ssl_prefer_server_ciphers on;
        ssl_session_cache shared:SSL:20m;
        ssl_session_timeout 180m;

        location / {
            proxy_pass http://localhost:3000/;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";
            proxy_set_header Host $http_host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto https;
            proxy_set_header X-Nginx-Proxy true;
            proxy_redirect off;
        }

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify
```

```
https://felipersantos.server:80/setup-wizard/1

  GNU nano 6.2
# HTTPS Server
    server {
        listen 80 ssl;
        server_name felipersantos.server;
```

We can also change which port our SSL server will listen in Nginx, that will make an easier to other devices access our localhost server if we want to.

Run this command to set permissions:

sudo chmod 400 /etc/nginx/ssl/feliperosario.key

And this one to Generate Strong Diffie Helman group:

sudo openssl dhparam -out /etc/nginx/dhparams.pem 2048

Next, restart Nginx and check it status:

sudo systemctl restart nginx

sudo systemctl status nginx

In the beginning of this guide, we cloned the composer.yml file, and now is the time to work with him.

Create these two directories below:

sudo mkdir -p /var/www/rocket.chat/data/runtime/db

sudo mkdir -p /var/www/rocket.chat/data/dump

Now, navigate to where the composer.yml file is located, and run this command to start the docker container:

sudo docker compose up -d

```
felipersantos@felipersantos:~/Desktop/Rocket Chat Felipe $ sudo docker compose u
p -d
[sudo] password for felipersantos:
[+] Running 12/12
 ✔ mongodb 1 layers [⣿]      0B/0B      Pulled                          24.9s
   ✔ 7d8255b8684c Pull complete                                         21.4s
 ✔ rocketchat 9 layers [⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿]      0B/0B      Pulled        60.3s
   ✔ 26c5c85e47da Pull complete                                          5.3s
   ✔ 96da4c1974ec Pull complete                                          5.3s
   ✔ 286584c9c618 Pull complete                                          8.3s
   ✔ ec51043fad6b Pull complete                                          8.5s
   ✔ 10845595c672 Pull complete                                          8.6s
   ✔ 88c93b4a0d5b Pull complete                                          9.3s
   ✔ de6b2b7fff59 Pull complete                                         53.6s
   ✔ 44b28869aeae Pull complete                                         55.5s
   ✔ 4f4fb700ef54 Pull complete                                         55.8s
[+] Building 0.0s (0/0)
[+] Running 4/4
 ✔ Network rocketchatfelipe_default         Created                     0.5s
 ✔ Volume "rocketchatfelipe_mongodb_data"   Created                     0.1s
 ✔ Container rocketchatfelipe-mongodb-1     Started                     3.3s
 ✔ Container rocketchatfelipe-rocketchat-1  Started                     0.8s
felipersantos@felipersantos:~/Desktop/Rocket Chat Felipe $
```

AUTOMATIC STARTUP & CRASH RECOVERY

Create the upstart job for MongoDB and its container:

sudo nano /etc/init/rocketchat_mongo.conf

Paste the following text:

(It's available to copy in https://docs.rocket.chat/deploy/prepare-for-your-deployment/rapid-deployment-methods/docker-and-docker-compose/docker-containers)

```
description "MongoDB service manager for rocketchat"

# Start MongoDB after docker is running
start on (started docker)
stop on runlevel [!2345]

# Automatically Respawn with finite limits
respawn
respawn limit 99 5

# Path to our app
chdir /var/www/rocket.chat

script
    # Showtime
    exec /usr/local/bin/docker compose up mongo
end script
```

Create the upstart job for Rocket.Chat and its container:

sudo nano /etc/init/rocketchat_app.conf

Paste the following text:

(It's available to copy/paste in https://docs.rocket.chat/deploy/prepare-for-your-deployment/rapid-deployment-methods/docker-and-docker-compose/docker-containers)

```
description "Rocket.Chat service manager"

# Start Rocket.Chat only after mongo job is running
start on (started rocketchat_mongo)
stop on runlevel [!2345]

# Automatically Respawn with finite limits
respawn
respawn limit 99 5

# Path to our app
chdir /var/www/rocket.chat

script
    # Bring up rocketchat app and hubot
    exec /usr/local/bin/docker compose up rocketchat hubot
end script
```

Now, reboot the OS:

sudo reboot

After the reboot, we use the command below to see if Docker containers are running in our environment:

sudo docker ps -a

```
felipersantos@felipersantos:~$ sudo docker ps -a
[sudo] password for felipersantos:
CONTAINER ID   IMAGE                                          COMMAND                  CREATED         STATUS          PORTS                    NAMES
ad19028941a6   registry.rocket.chat/rocketchat/rocket.chat:latest   "docker-entrypoint.s…"   3 minutes ago   Up 32 seconds   0.0.0.0:3000->3000/tcp   rocketchatfelipe-rocketchat-1
021cef07a656   bitnami/mongodb:5.0                            "/opt/bitnami/script…"   3 minutes ago   Up 32 seconds   27017/tcp                rocketchatfelipe-mongodb-1
```

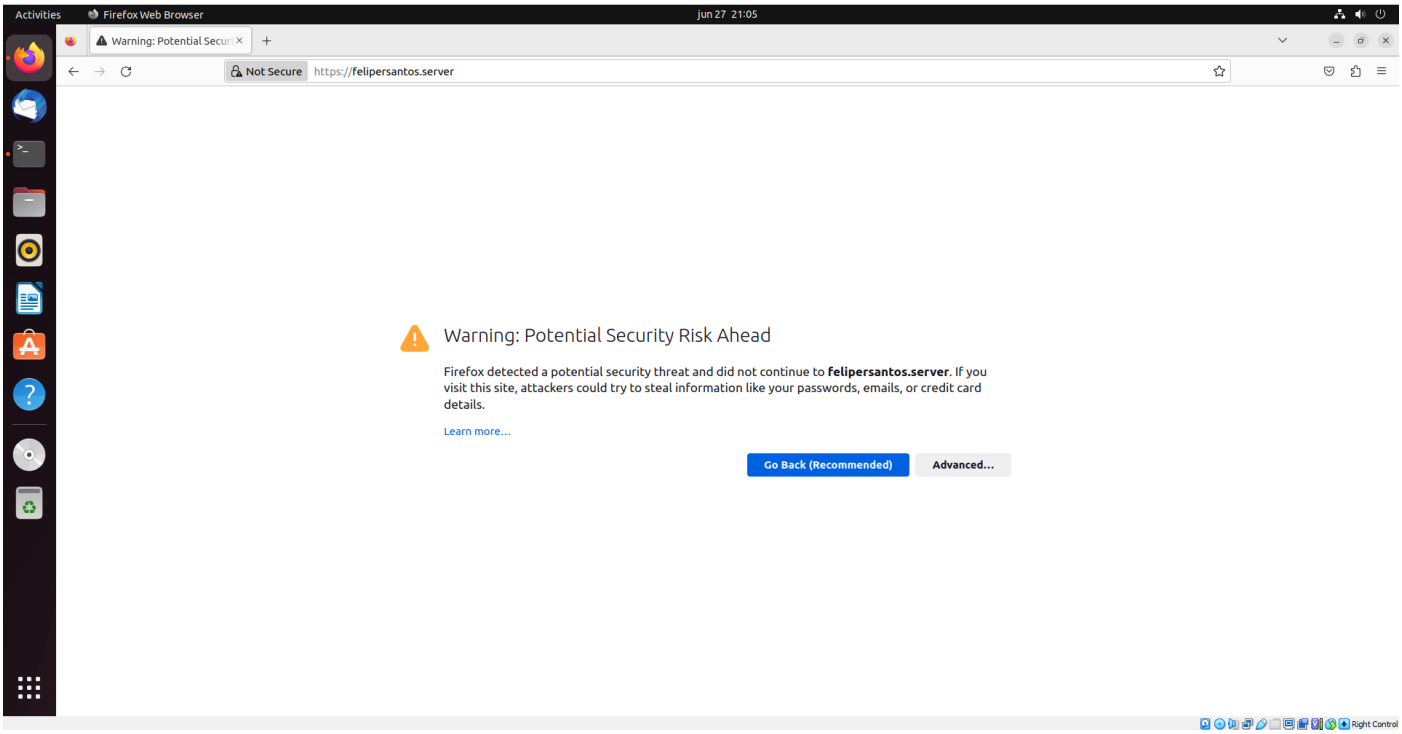
If all appears to be working, it's time to see if it's working in a browser, accessing the localhost:

(I'm using my local domain, as I configured in the steps before)

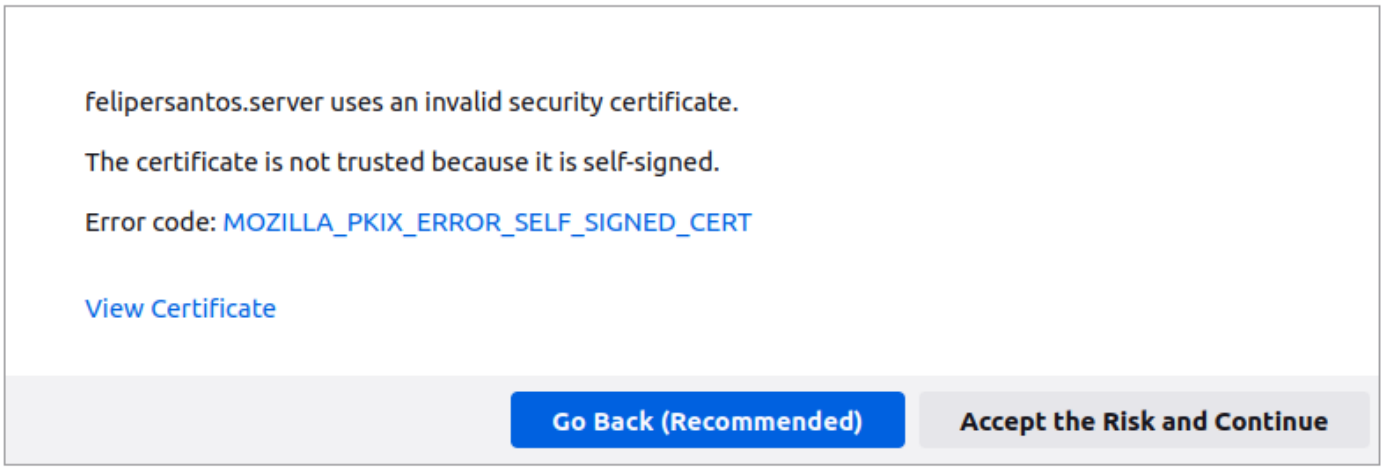
```
felipersantos@felipersantos: ~
  GNU nano 6.2                                                    /etc/hosts
127.0.0.1       localhost
127.0.1.1       felipersantos.server    felipersantos

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

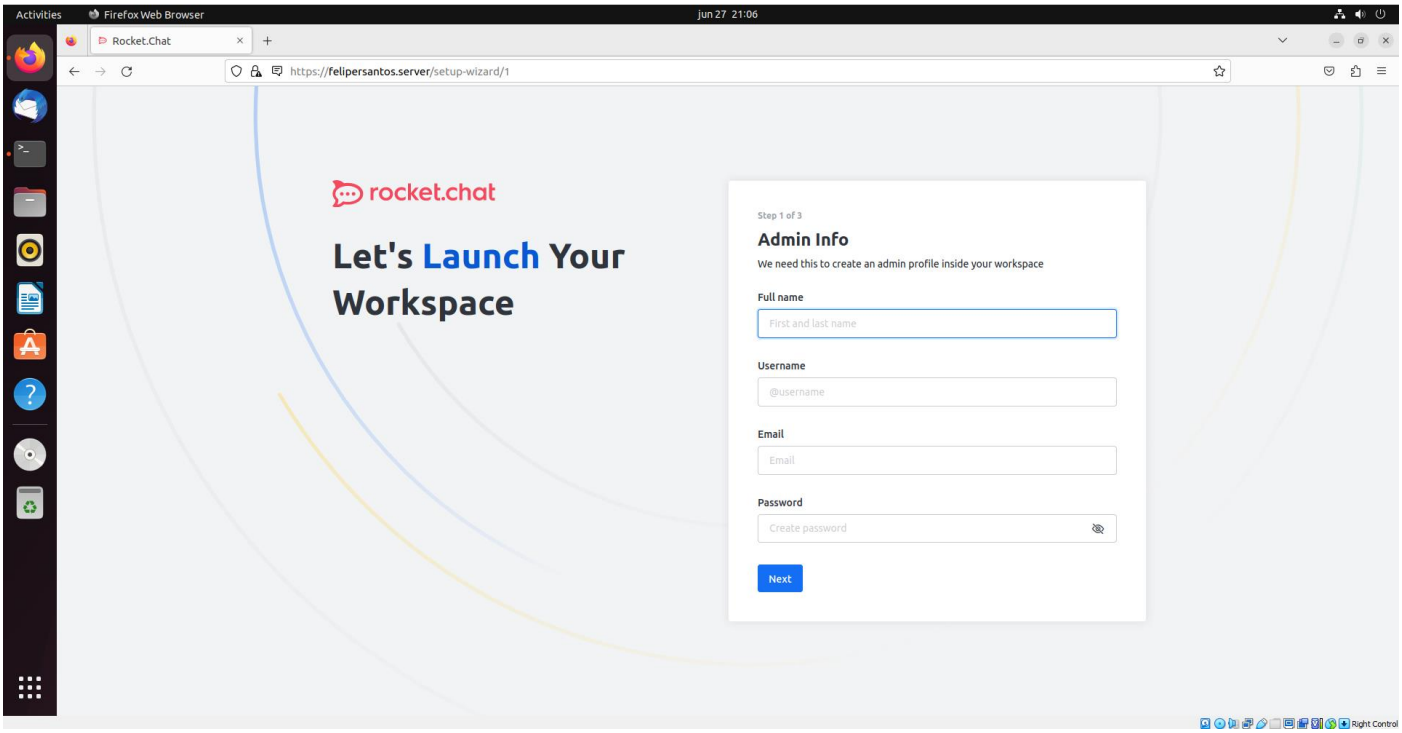
Now on the address field on our browser, we are going to test if the HTTPS protocol is working as well:

https://felipersantos.server



The browser shows that message cause we're using a self-signed SSL certificate, it will not appear if you have a certificate signed by an authority, we can click on "Advanced…" and accept the risk and continue:





If we did our deploy correctly, we should now see the Rocket.Chat setup wizard screen, which means that all of our settings are working just fine.

EXPOSING OUR LOCAL HOST TO THE INTERNET

Now that our Rocket.Chat website is working, we'll let the staff access it to evaluate the challenge, we can use either Ngrox or serveo.net ssh tool.

Basically, Ngrok is a cross-platform application that enables developers to expose a local development server to the Internet with minimal effort. The software makes your locally-hosted web server appear to be hosted on a subdomain of ngrok.com.

Serveo is an SSH server just for remote port forwarding. When a user connects to Serveo, they get a public URL that anybody can use to connect to their localhost server.

For use Ngrox, install Ngrok's snap using this command on the terminal:

snap install ngrok

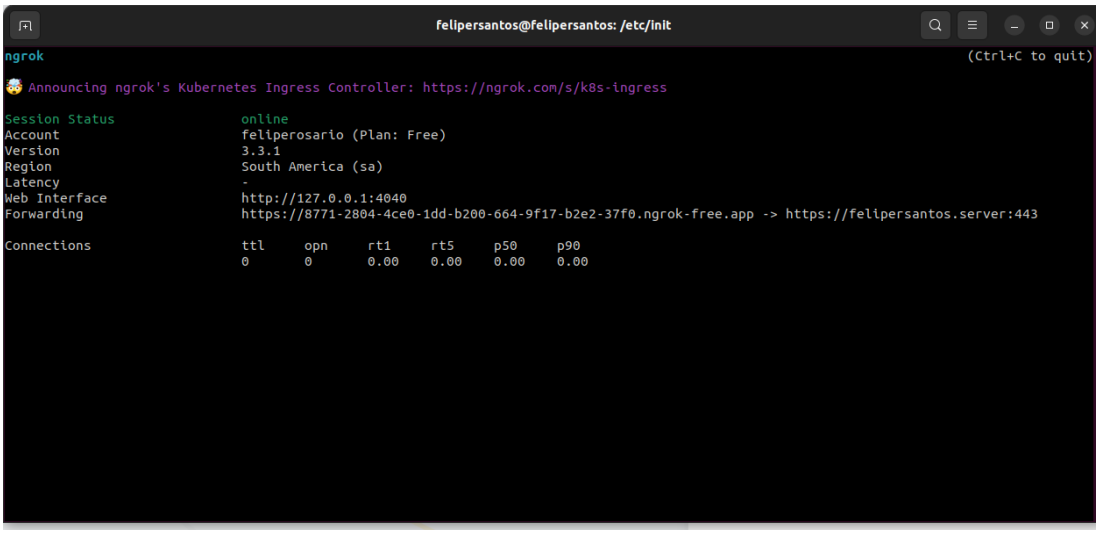
We can check if it is installed using the command ngrok –version

Before running ngrok, we have to set our authtoken, you can get it from sign up in their website.

ngrok config add-authtoken <auth-token>

There are a lot of configurations you can do with Ngrok, but for this test, we're doing it in a simple way:

ngrok http https://felipersantos.server:443

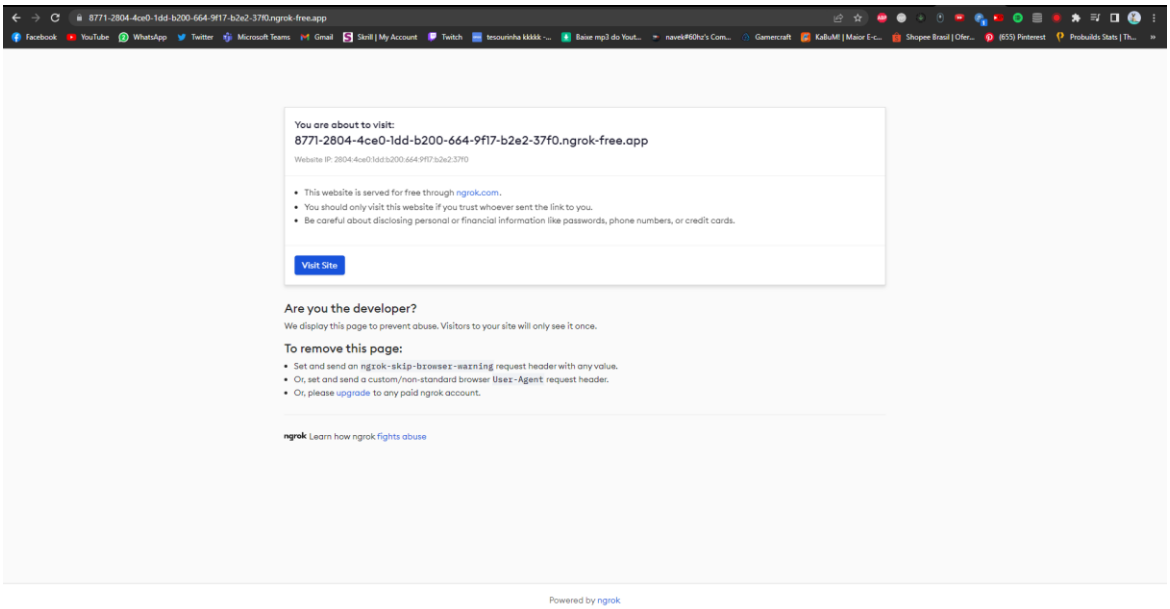


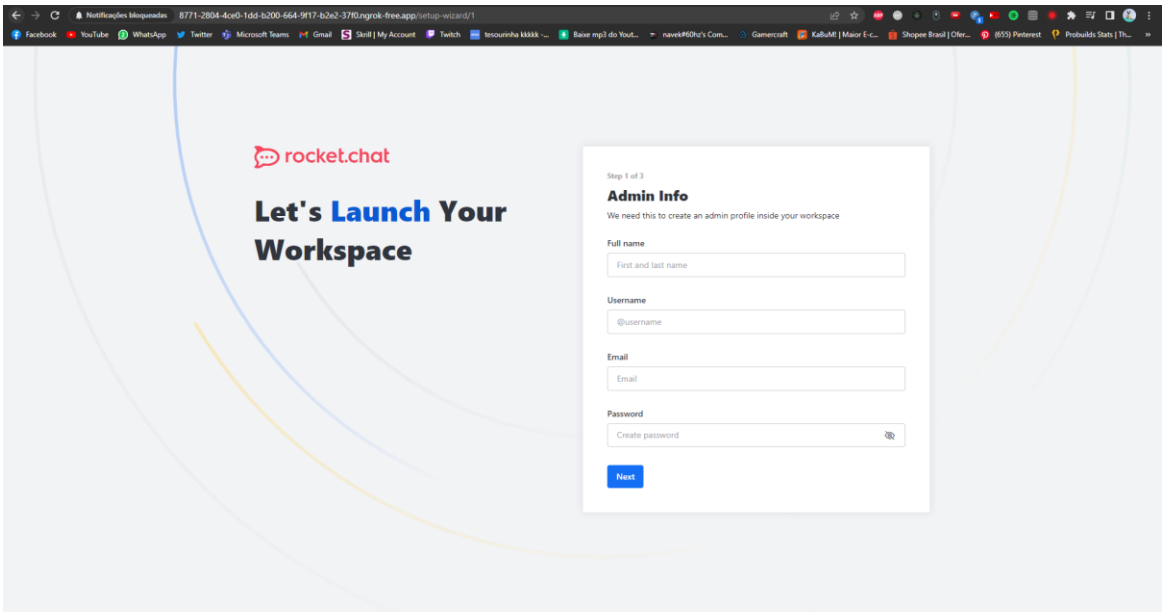
We should now see this terminal instance above, the forward link, is the one we're sharing with the anyone who wants to access our website:

https://8771-2804-4ce0-1dd-b200-664-9f17-b2e2-37f0.ngrok-free.app

You can test if it's working, by accessing this same link in another device.

This warning below is telling that the website we're visiting is hosted on Ngrok servers, click on Visit Site, and we'll be accessing the local host we published.





We can also use Serveo to expose our localhost:

Serveo is a lot simpler than Ngrox to use:

ssh -R 80:feliperosario.server:80 serveo.net

The -R option instructs your SSH client to request port forwarding from the server and proxy requests to the specified host and port (usually localhost). A subdomain of serveo.net will be assigned to forward HTTP traffic.

But, for doing that and not get bad requests or forward protocol errors, cause we're using HTTPS, we need to change some lines on our default config file:

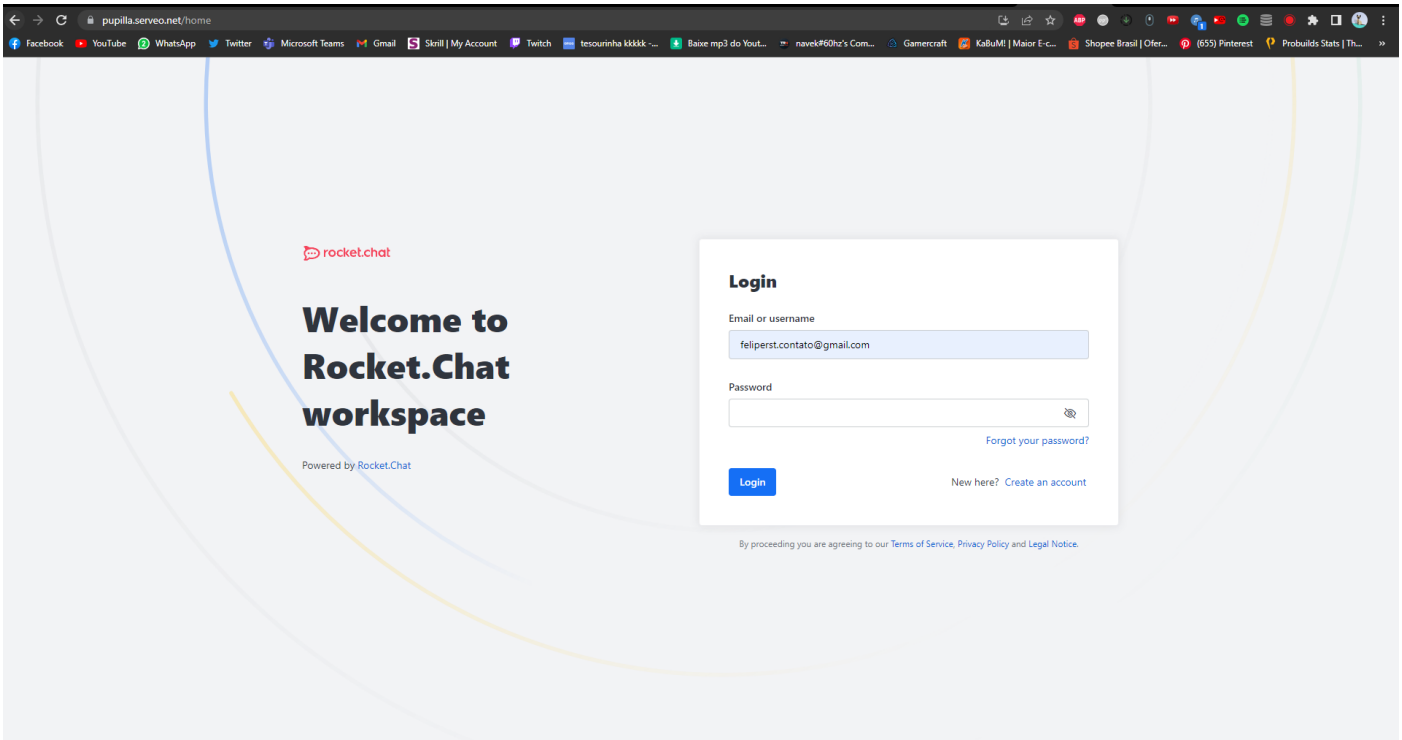
sudo nano /etc/nginx/sites-available/default

```
  GNU nano 6.2                    /etc/nginx/sites-available/default
server {
        listen 80 default_server;
        listen [::]:80 default_server;
        server_name felipersantos.server;

        location / {
             proxy_pass http://felipersantos.server:3000;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";
            proxy_set_header Host $http_host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto http;
            proxy_set_header X-Nginx-Proxy true;
            proxy_redirect off;
        }
}

server {
        listen 443 ssl;
        server_name felipersantos.server;

        error_log /var/log/nginx/rocketchat_error.log;
        ssl off;
        ssl_certificate /etc/nginx/ssl/feliperosario.crt;
        ssl_certificate_key /etc/nginx/ssl/feliperosario.key;
        ssl_dhparam /etc/nginx/dhparams.pem;
        ssl_protocols TLSv1.2;
        ssl_ciphers 'ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES>
        ssl_prefer_server_ciphers on;
        ssl_session_cache shared:SSL:20m;
        ssl_session_timeout 180m;

        location / {
            proxy_pass http://felipersantos.server:3000;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";
            proxy_set_header Host $http_host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy
```

We need to add another server "{}" key, to separate the SSL https server of the HTTP listening server, and change the port and the forward protocol on the proxy_set_header. Don't worry, Serveo and Ngrox still let us use a secure connection.

The subdomain provided by Serveo is based on your Ip address, and it may change:

https://pupilla.serveo.net



Admin login to this Rocket.chat server:

Username: felipersantos

Password: rocketchatchallenge@2023

INTEGRATING OUR APPLICATION WITH GITHUB (OPTINAL)

We can follow this guide below to do our Github and Rocket.Chat server integration:

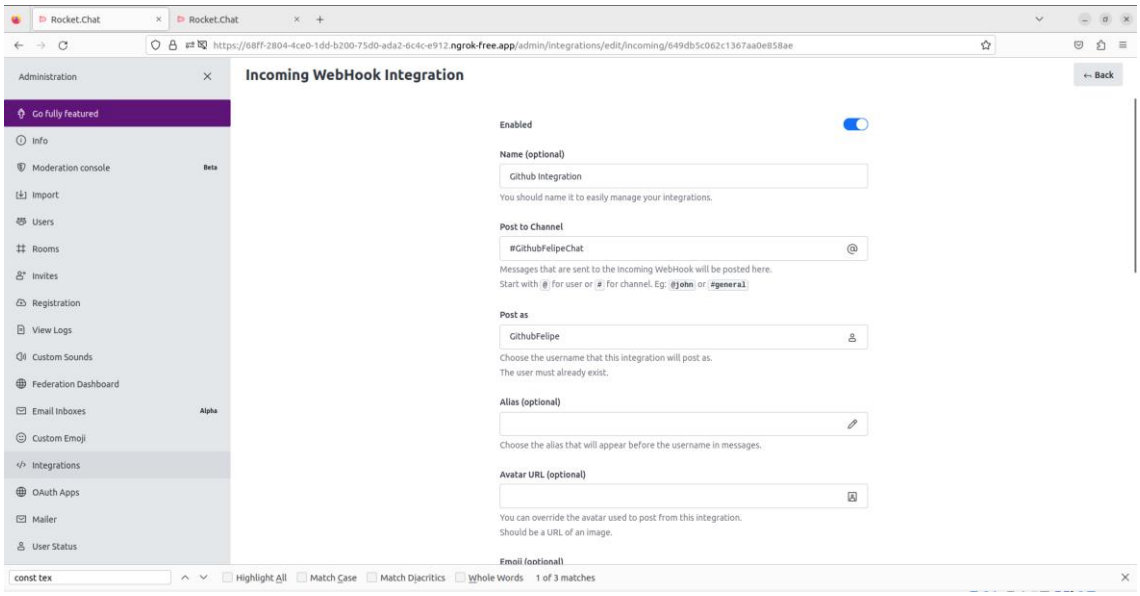
https://docs.rocket.chat/use-rocket.chat/workspace-administration/integrations/github

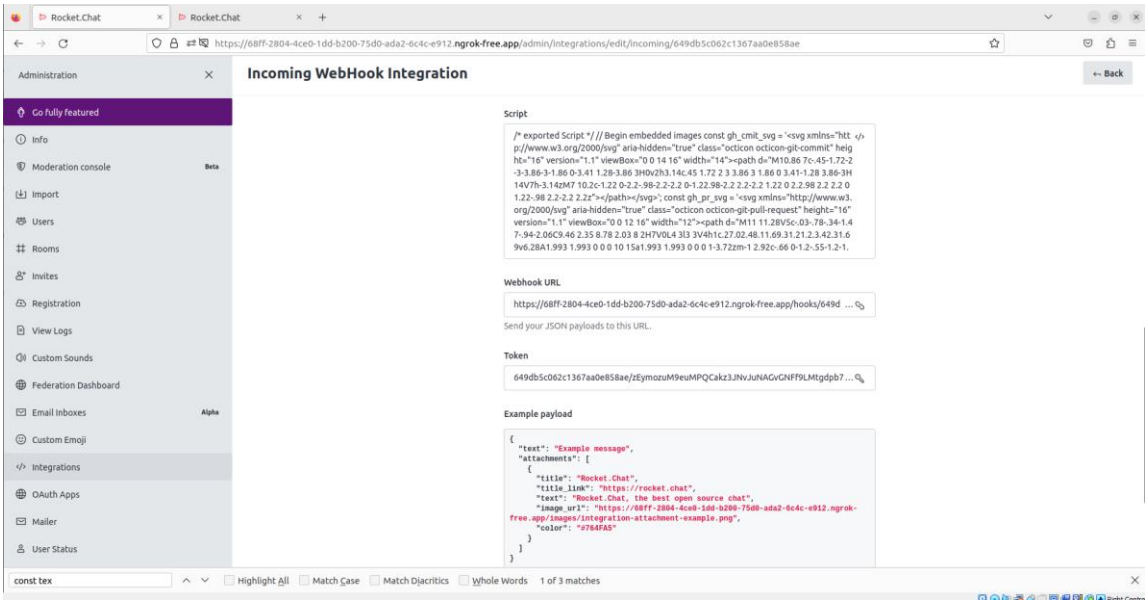
Creating new GitHub webhook integration:

Go to the Administration -> Workspace -> Integrations settings on your Rocket.Chat workspace

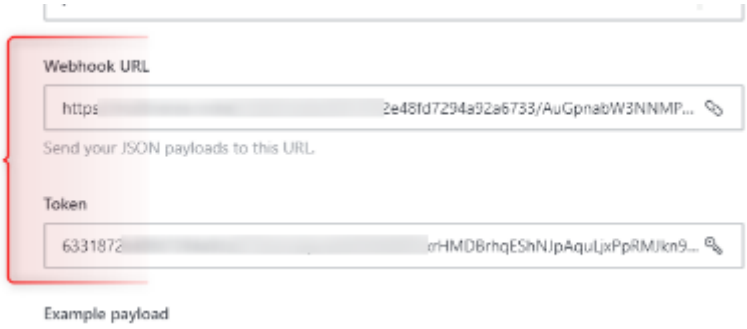
Switch to the Incoming tab and create a New Integration

Fill in the details of your webhook including the name of the webhook, the room to post into, the user to post as and enable it, than paste the script on the link above, and save it.





We also need to copy these two fields:



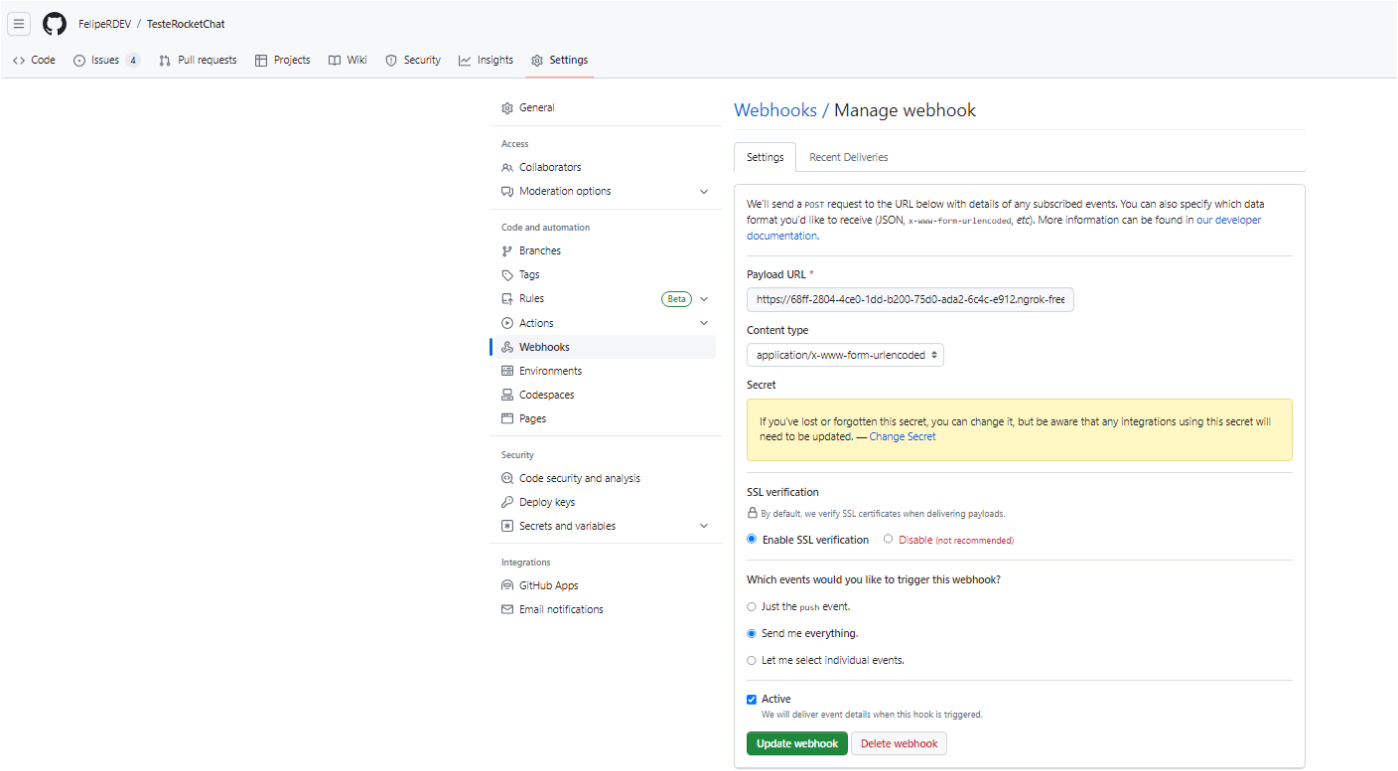
GitHub Webhook Settings:

After creating the new incoming webhook integration on Rocket.Chat, it is time to link it up with the GitHub repository.

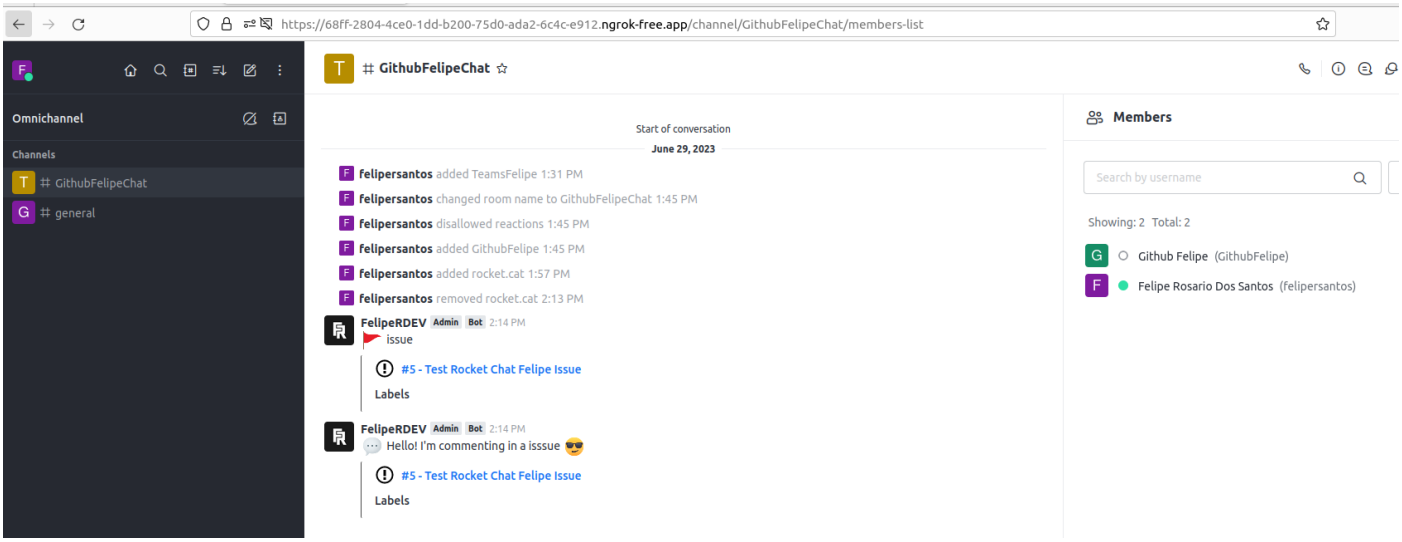
Go to the GitHub project repository then navigate to Settings > Webhooks

Add webhook and fill in the URL and token you copied from the Rocket.Chat setting

Select the list of events you want to be notified on and Add webhook



After successful configuration, you can test the Webhook with any event trigger and see the notification in your specified Rocket.Chat room.



This is just a simple example of how we can setup integrations in Rocket.Chat using webhooks, there are a lot of other webhook integrations available at:

https://docs.rocket.chat/use-rocket.chat/workspace-administration/integrations

API TESTS

For API tests we're using Postman

sudo snap install postman

All the API test results are available in this Github link:

https://github.com/FelipeRDEV/Rocket.Chat_ES_Felipe_Santos

This section will not be detailed as the previous ones, cause the focus here is to document the results of the requests.

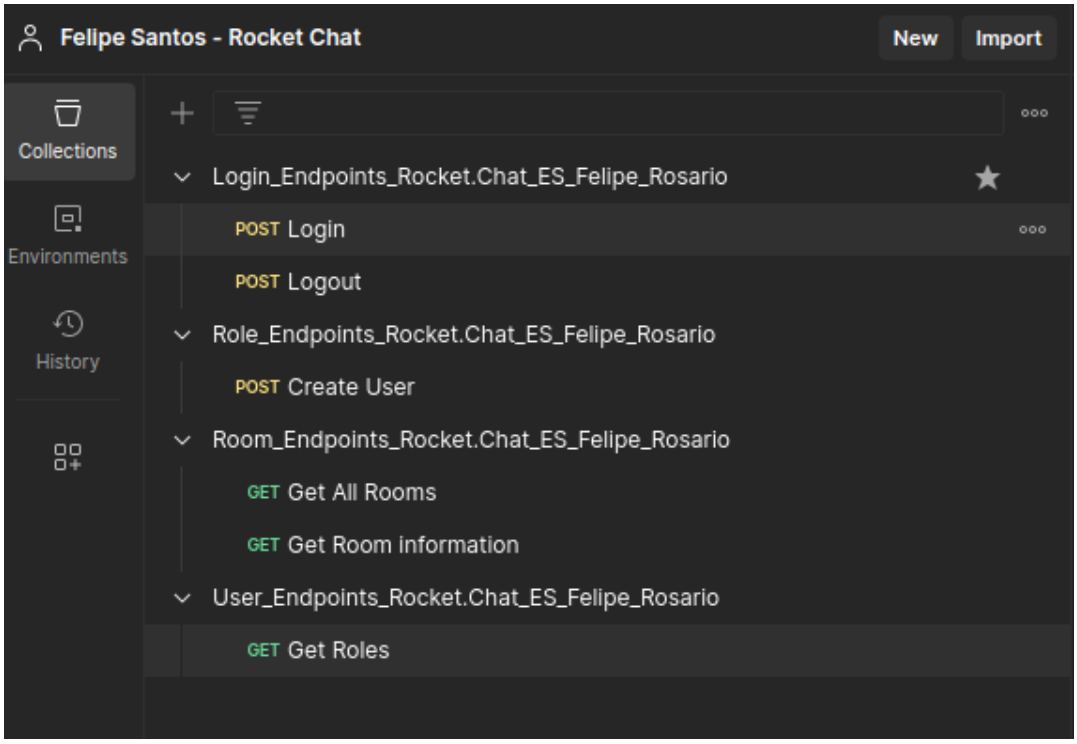
Note: the server address changes every time the application runs (ngrok) , so be aware that you got to change address on the request fields in every postman collection.

The picture below show's all the postman collections created to do the following tasks:

1. Create a new user via an API endpoint

2. Get the room information via an API endpoint

3. Get a list of all user roles in the system via an API endpoint



Login

POST Login | Overview | Rocket_Chat_Felipe_R

Login_Endpoints_Rocket.Chat_ES_Felipe_Rosario / **Login**

Save

POST https://7eba-2804-4ce0-1dd-b200-b850-847c-d3e4-3bfb.ngrok-free.app/api/v1/login Send

Params | Authorization | Headers (8) | Body | Pre-request Script | Tests | Settings | Cookies

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL | JSON | Beautify

```
{
    "user": "felipersantos",
    "password": "rocketchatchallenge@2023"
}
```

Body | Cookies | Headers (19) | Test Results | Status: 200 OK | Time: 114 ms | Size: 2.3 KB | Save as Example

Pretty | Raw | Preview | Visualize | JSON


```
{
    "status": "success",
    "data": {
        "userId": "Hp4BjCSxjBKznCHX3",
        "authToken": "2GXyeByoV_Ww0VkHk6TaNt91mb3awAxfTR7lfq7KhWB",
        "me": {
            "_id": "Hp4BjCSxjBKznCHX3",
            "services": {
                "password": {
                    "bcrypt": "$2b$10$Orn7n.H4Bmt0z82kpNyMOeXC3jOg2FDu1kAYPFjUfespTMYABCxM2"
                },
                "email2fa": {
                    "enabled": true
                }
```


Runner | Capture requests | Cookies | Trash

When the request is made, it sets an global variable value to the authToken value, so we don't need to type it every time we make a request:

Login_Endpoints_Rocket.Chat_ES_Felipe_Rosario / **Login**

POST https://7eba-2804-4ce0-1dd-b200-b850-847c-d3e4-3bfb.ngrok-free.app/api/v1/login

Params | Authorization | Headers (8) | Body | Pre-request Script | Tests | Settings

```
pm.globals.set ('token', pm.response.json().data.authToken)
```

Globals Edit

Variable	Initial value	Current value
token		IOsQyndSpjL5RtOs3Wv2Ziw3NC_mDXPtACDAMQEI7B0

The logout request was not mandatory, but was made just for a test.

Create a user

We need to be logged in to create a user, like we did before, now we'll use the authToken variable and our user Id to make this request:

User_Endpoints_Rocket.Chat_ES_Felipe_Rosario / **Create User**

Save

POST https://7eba-2804-4ce0-1dd-b200-b850-847c-d3e4-3bfb.ngrok-free.app/api/v1/users.create Send

Params Authorization Headers (11) Body Pre-request Script Tests Settings Cookies

Headers 9 hidden

Key	Value	Description
X-User-Id	Hp4BjCSxjBKznCHX3	
X-Auth-Token	{{token}}	
Key	Value	Description

Bulk Edit Presets

User_Endpoints_Rocket.Chat_ES_Felipe_Rosario / **Create User**

Save

POST https://7eba-2804-4ce0-1dd-b200-b850-847c-d3e4-3bfb.ngrok-free.app/api/v1/users.create Send

Params Authorization Headers (11) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2     "name": "TestFelipe3",
3     "email": "TesteFelipe3@user.numberone",
4     "password": "RocketChallenge1",
5     "username": "TestFelipe3"
6 }
```

User_Endpoints_Rocket.Chat_ES_Felipe_Rosario / **Create User**

Save

POST https://7eba-2804-4ce0-1dd-b200-b850-847c-d3e4-3bfb.ngrok-free.app/api/v1/users.create Send

Params Authorization Headers (11) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2     "name": "TestFelipe3",
3     "email": "TesteFelipe3@user.numberone",
4     "password": "RocketChallenge1",
5     "username": "TestFelipe3"
6 }
```

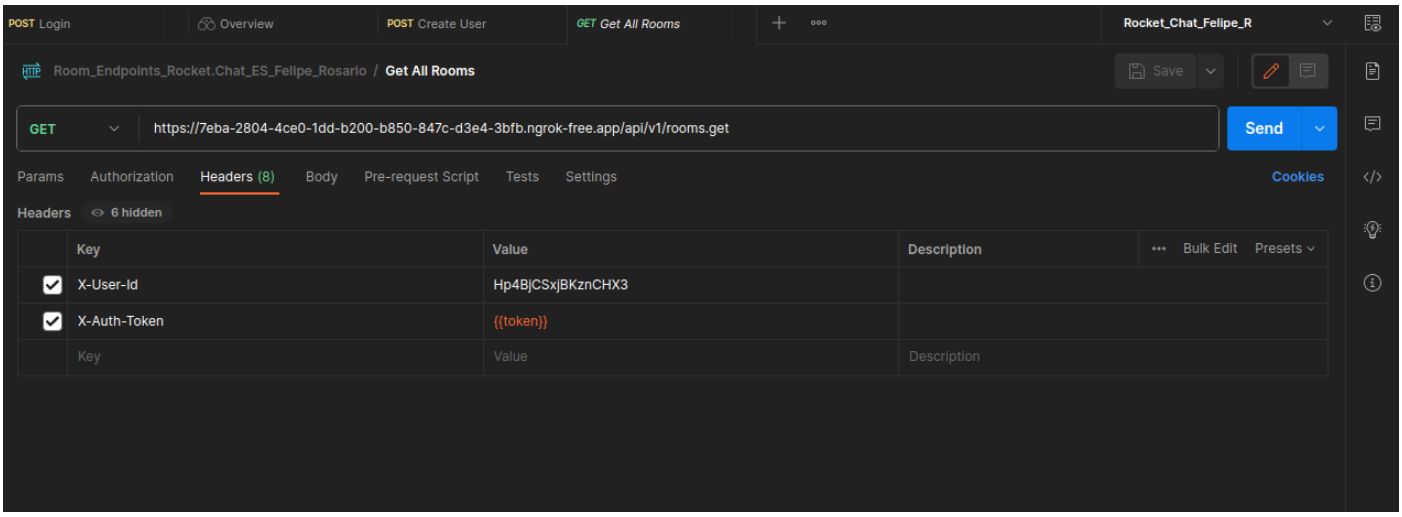
Body Cookies Headers (16) Test Results Status: 200 OK Time: 357 ms Size: 1.14 KB Save as Example

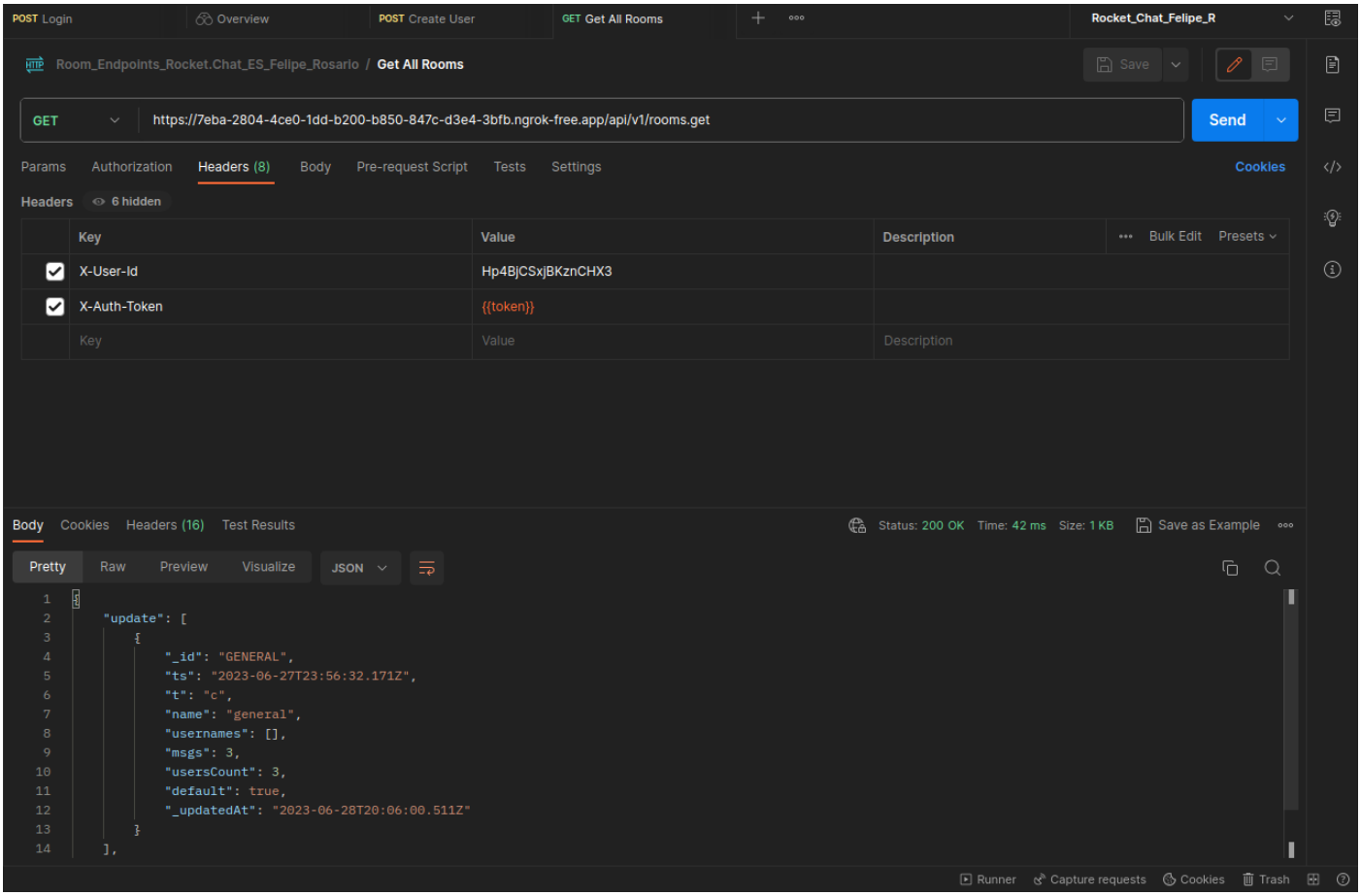
Pretty Raw Preview Visualize JSON

```
1 {
2     "user": {
3         "_id": "aFHrkDzBrh5jDzvYH",
4         "createdAt": "2023-06-28T20:13:03.870Z",
5         "username": "TestFelipe3",
6         "emails": [
7             {
8                 "address": "TesteFelipe3@user.numberone",
9                 "verified": false
10            }
11        ],
12        "type": "user",
13        "status": "offline",
14        "active": true,
```

Get Room information

I've made a request to get all rooms, so we can put a specific room of all the rooms, to see informations about it:





Now we can grab the id and the name of a room, and see its information, using the request below:

Room_Endpoints_Rocket.Chat_ES_Felipe_Rosario / **Get Room information**

Save

GET https://7eba-2804-4ce0-1dd-b200-b850-847c-d3e4-3bfb.ngrok-free.app/api/v1/rooms.info?roomId=GENERAL&roomName=general

Send

Params • Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Headers 6 hidden

	Key	Value	Description
☑	X-User-Id	Hp4BjCSxjBKznCHX3	
☑	X-Auth-Token	{{token}}	
	Key	Value	Description

Bulk Edit Presets

POST Login | Overview | POST Create User | GET Get All Rooms | GET Get Room information | Rocket_Chat_Felipe_R

Room_Endpoints_Rocket.Chat_ES_Felipe_Rosario / **Get Room information**

Save

GET https://7eba-2804-4ce0-1dd-b200-b850-847c-d3e4-3bfb.ngrok-free.app/api/v1/rooms.info?roomId=GENERAL&roomName=general

Send

Params • Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

	Key	Value	Description
☑	roomId	GENERAL	
☑	roomName	general	
	Key	Value	Description

Bulk Edit

Body Cookies Headers (16) Test Results

Status: 200 OK Time: 43 ms Size: 1013 B Save as Example

Pretty Raw Preview Visualize JSON


```
{
    "room": {
        "_id": "GENERAL",
        "ts": "2023-06-27T23:56:32.171Z",
        "t": "c",
        "name": "general",
        "usernames": [],
        "msgs": 3,
        "usersCount": 3,
        "default": true,
        "_updatedAt": "2023-06-28T20:06:00.511Z"
    },
    "success": true
}
```


Runner Capture requests Cookies Trash

Get All User Roles in the system

Role_Endpoints_Rocket.Chat_ES_Felipe_Rosario / **Get Roles**

Save

GET https://7eba-2804-4ce0-1dd-b200-b850-847c-d3e4-3bfb.ngrok-free.app/api/v1/roles.list Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Headers 7 hidden

	Key	Value	Description
☑	X-User-Id	Hp4BjCSxjBKznCHX3	
☑	X-Auth-Token	{{token}}	
	Key	Value	Description

Bulk Edit Presets

Body Cookies Headers (16) Test Results Status: 200 OK Time: 41 ms Size: 2.05 KB Save as Example

Pretty Raw Preview Visualize JSON


```
{
    "roles": [
        {
            "_id": "admin",
            "scope": "Users",
            "description": "Admin",
            "mandatory2fa": false,
            "name": "admin",
            "protected": true
        },
        {
            "_id": "moderator",
            "scope": "Subscriptions",
            "description": "Moderator",
```


Runner Capture requests Cookies Trash