Technical Assessment

Rocket.Chat Enterprise Solutions Team

Technical Support Analyst

Candidate: Rafael Pinto Dias

Campinas - Brazil, february de 2022

1 - Deploying Rocket.Chat

In this section all steps from Rocket.Chat deployment will be commented on.

1.1 - Hardware Requirements

The server was based on a computer with limited hardware, that is above:

- Processor: Intel(R) Core (TM) i5-3337U CPU @ 1.80GHz
- CPU: 4
- Memory: 6GB
- HDD: 1TB

The operating system was virtualized by the VirtualBox software. The distribution Ubuntu 18.04.6 (Linux) was used. Below are the VM hardware specifications:

- Operating System: Ubuntu 64 bits (Linux)
- Memory: 2542 MB
- Network: Bridge mode.
- Storage: 50 GB

1.2 - Deploy Method

The deploy method chosen was the Docker & Docker Compose because of the simplicity and familiarity with this method.

First I installed Docker to the server then the docker-compose.yml was downloaded directly by executing the following command.

curl -L https://go.rocket.chat/i/docker-compose.yml -O

After this the container can be start up by executing:

docker-compose up -d

To verify that everything went well and is working, we can run the command:

docker ps

And will return the following containers and it's status.

2 - APIs Test

In this section all APIs tests will be explained and shown.

I was supposed to run the following queries/updates:

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

2.1 - Create a new user via an API endpoint

Before creating a user via an API endpoint, we need to generate a token and a user-ID. This user must have permissions to create logins.

Both tokens can be generating by the panel:

Accessing Profile -> My Account -> Security -> Personal Access Tokens

After generating personal access tokens, the call can be done.

```
curl -H "X-Auth-Token: XYrSWDNKZOm1G1BqHVWvOP91J7YQKas430wIrhjgvhx" \
    -H "X-User-Id: Tb6gzsAfeMC57b4Qy" \
    -H "Content-type:application/json" \
    http://192.168.13.105:3000/api/v1/users.create \
    -d '{"name": "Rodrigo Cruz", "email": "rdc@rockechat", "password": "senhaboa", "username": "rdc"}'
```

Note: The IP 192.168.13.105:3000 is from the server local network interface. To make the call over another network, it will be necessary to change the line to *https://rafaelchat.loca.lt//api/v1/users.create* \

As response


```
{
  "user":{
    "_id":"jjYunsiBZgmipt9mB",
    "createdAt":"2022-02-14T06:26:09.252Z",
    "username":"rdc",
    "emails":[
      {
        "address":"rdc@rockechat",
        "verified":false
      }
    ],
    "type":"user",
    "status":"offline",
    "active":true,
    "_updatedAt":"2022-02-14T06:26:09.565Z",
    "__rooms":[
      "GENERAL"
    ],
    "roles":[
      "user"
    ],
    "name":"Rodrigo Cruz",
    "settings":{

    }
```

```
  },
  "success":true
}
```

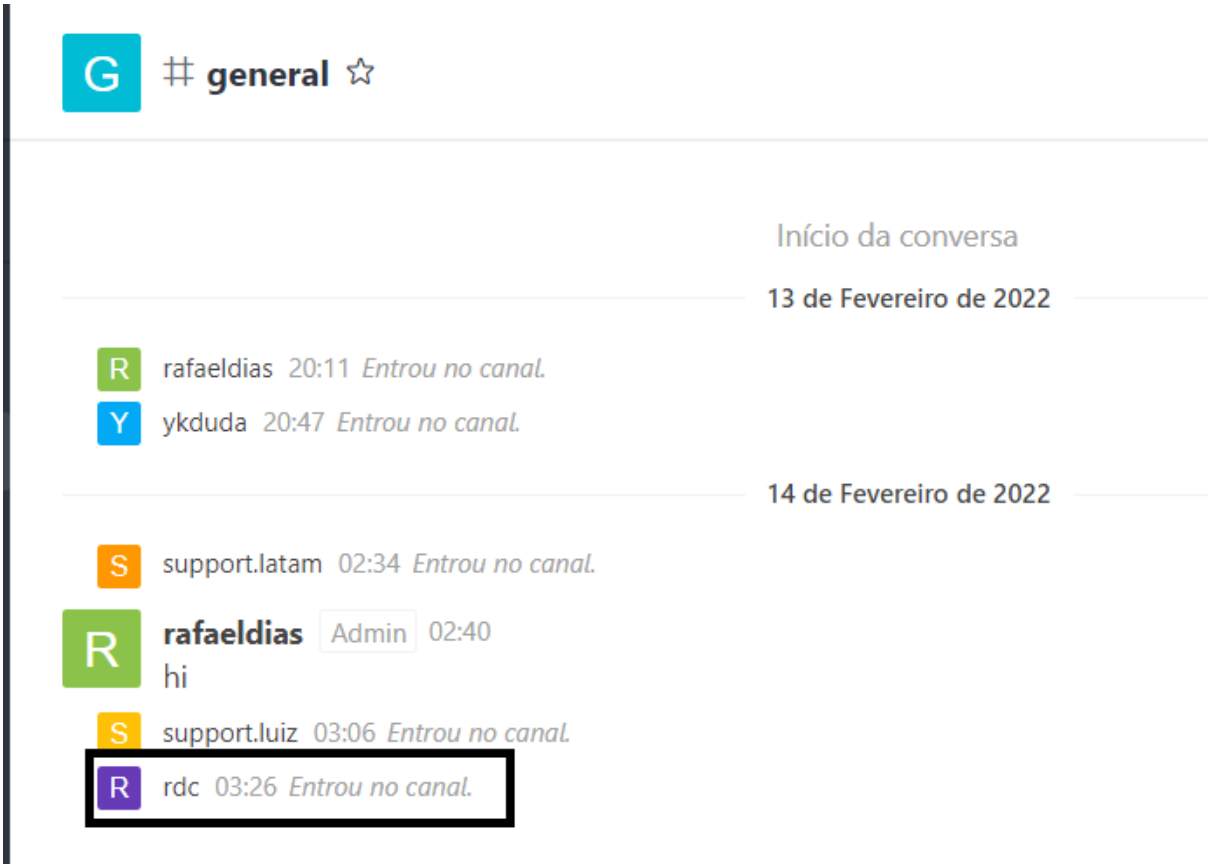

Figure 1 - User Create

2.2 - Get the room information via an API endpoint

Using the tokens generated in the Section 2.1 we can use the call below to see all information about some room or channel.

```
curl -H "X-Auth-Token: XYrSWDNKZOm1G1BqHVWvOP91J7YQKas430wIrhjgvhx" \
>     -H "X-User-Id: Tb6gzsAfeMC57b4Qy" \
>     http://192.168.13.105:3000/api/v1/rooms.info?roomId=4qfRF4sNiRwT4rujT
```

As response

```
{
  "room":{
    "_id":"4qfRF4sNiRwT4rujT",
    "fname":"Team-Support",
    "customFields":{

    },
    "description":"Support Stuff",
```

```
    "broadcast":false,
    "encrypted":false,
    "name":"Team-Support",
    "t":"c",
    "msgs":3,
    "usersCount":2,
    "u":{
      "_id":"Tb6gzsAfeMC57b4Qy",
      "username":"rafaeldias"
    },
    "ts":"2022-02-14T05:41:46.091Z",
    "ro":false,
    "default":false,
    "sysMes":true,
    "_updatedAt":"2022-02-14T05:55:16.666Z",
    "lastMessage":{
      "_id":"3Xzh7KF8HYQ9fMkEr",
      "rid":"4qfRF4sNiRwT4rujT",
      "msg":"ola",
      "ts":"2022-02-14T05:55:16.544Z",
      "u":{
        "_id":"Tb6gzsAfeMC57b4Qy",
        "username":"rafaeldias",
        "name":"Rafael Dias"
      },
      "_updatedAt":"2022-02-14T05:55:16.650Z",
      "urls":[

      ],
      "mentions":[

      ],
      "channels":[

      ],
      "md":[
        {
          "type":"PARAGRAPH",
          "value":[
            {
              "type":"PLAIN_TEXT",
              "value":"ola"
            }
          ]
        }
      ]
    },
    "lm":"2022-02-14T05:55:16.544Z"
  },
  "success":true
}
```

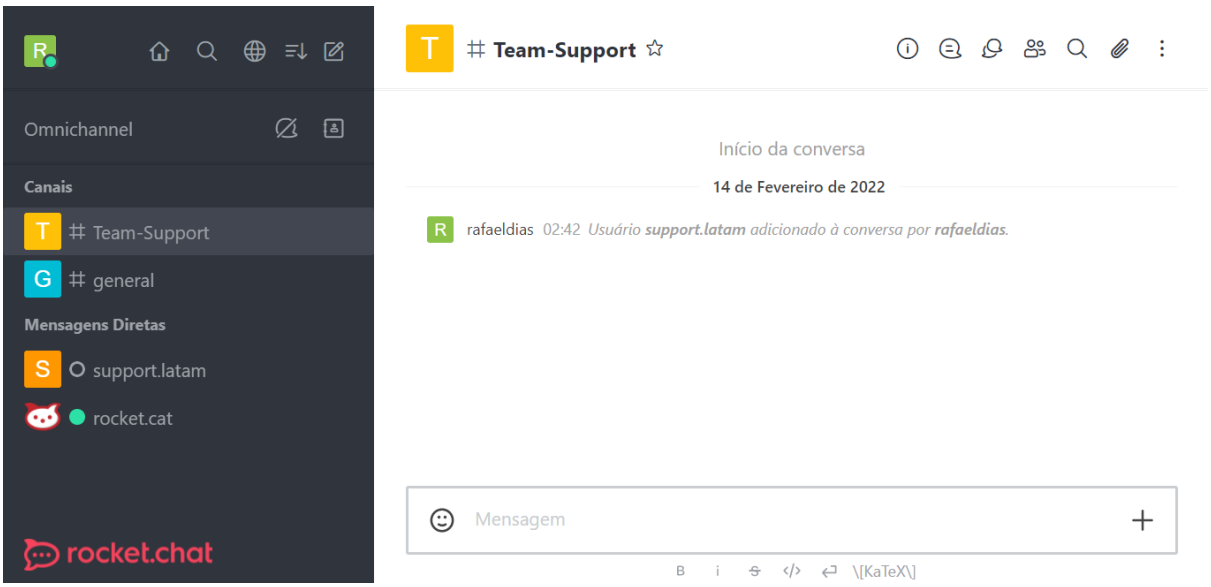

Figure 2 - Room Team Support

2.3 - Get a list of all user roles in the system via an API endpoint

Using the tokens generated in the Section 2.1 we can use the call below to see all users created.

```
curl -H "X-Auth-Token: XYrSWDNKZOm1G1BqHVWvOP91J7YQKas430wIrhjgvhx" \
>     -H "X-User-Id: Tb6gzsAfeMC57b4Qy" \
>     http://192.168.13.105:3000/api/v1/users.list
```

As response:


```
{
  "users":[
    {
      "_id":"Tb6gzsAfeMC57b4Qy",
      "emails":[
        {
          "address":"rafael@rocket-chat.com",
          "verified":false
        }
      ],
      "status":"online",
      "active":true,
      "roles":[
        "user",
        "admin"
      ],
      "name":"Rafael Dias",
      "lastLogin":"2022-02-14T05:33:55.637Z",
      "username":"rafaeldias",
      "nameInsensitive":"rafael dias"
    },
    {
```

```
      "_id":"rocket.cat",
      "name":"Rocket.Cat",
      "username":"rocket.cat",
      "status":"online",
      "active":true,
      "roles":[
        "bot"
      ],
      "avatarETag":null,
      "nameInsensitive":"rocket.cat"
    },
    {
      "_id":"inYzBGtF776MgNh4Z",
      "username":"support.latam",
      "emails":[
        {
          "address":"support@rocket-chat",
          "verified":false
        }
      ],
      "status":"offline",
      "active":true,
      "roles":[
        "user"
      ],
      "name":"Rafael Support",
      "nameInsensitive":"rafael support"
    },
    {
      "_id":"n35huxCp8yoRQr5vp",
      "username":"ykduda",
      "emails":[
        {
          "address":"duda@rocket-chat",
          "verified":false
        }
      ],
      "status":"offline",
      "active":true,
      "roles":[
        "user"
      ],
      "name":"Eduarda",
      "nameInsensitive":"eduarda"
    }
  ],
  "count":4,
  "offset":0,
  "total":4,
  "success":true
}
```


3 - Internet Exposing

To expose the localhost to the internet using Localtunnel, I used the following command after installing as shown in the website [http://localtunnel.github.io/www]

PORT=3000 lt -s rafaelchat 192.168.13.105.

So the server can be accessed from any place by the link: https://rafaelchat.loca.lt/

Localtunnel allows you to easily share a web service on your local development machine without messing with DNS and firewall settings.

Localtunnel will assign you a unique publicly accessible url that will proxy all requests to your locally running web server