RocketChat

Fabrice Prigent

Université Toulouse 1 Capitole

Jeudi 2 juillet 2020

Escroc !!

Escroc !!!
Et le cluster BBB ?

Et le cluster BBB ?

- Il est en préparation, mais...
- Les priorités de l'établissement ne sont pas forcément là (Ah... Zoom....)
- Les priorités de l'équipe système changent,
  - Avec l'apparition de plus en plus d'outils en Docker
    - GreenLight,
    - Scalelite,
    - Guacamole,
    - BigBlueButton,
    - etc.
  - d'où l'intérêt de Kubernetes.
    - Et Kubernetes c'est compliqué.
- Et je déteste parler de quelque chose qui n'est pas en production.

Mais pourquoi RocketChat ?

Mais pourquoi RocketChat ?

Le confinement

- Besoin, réel ou supposé, de communication "temps réel" avec les personnels
- Achat d'un produit (Whaller) de 10 K€pour 200 personnes pour 1 an par la direction
- Le chat est "ce qui se fait".
- Mais
  - L'outil n'est pas intégré du tout (mot de passe...)
  - Les groupes sont manuels
  - Limité par le coût.
  - Volonté de la DSI d'avoir une offre de service "chat".

Les candidats

Les payants

- Slack
- Teams

Les gratuits

- XMPP
- Mattermost
- RocketChat
- Autres outils.

XMPP

- Protocole, plus qu'un outil
- Déjà disponible (mais pas utilisé) en local
- Intégré dans Thunderbird
- Utilisable pour des communications inter-processus.

Mais

- Pas sexy (pas d'intégration directe d'images alors que c'est trop bien.).

Mattermost

- Outil libre de référence.
- Nombreuses fonctionnalités et intégrations avec plusieurs outils.

Mais

- N'avait pas le vent en poupe

RocketChat

- Initialement, module commercial de Konecty créé par Gabriel Engel.
- Outil libre depuis 2015,
- Nombreuses fonctionnalités et intégrations avec plusieurs outils
- Outil sexy.

RocketChat

- https://www.rocket.chat
- Actuellement en version 3.4.0.
- Utilisable en SaaS ou en local,
- Version communautaire ou version entreprise,
- Client mobile possible (mais plutôt orienté SaaS).

RocketChat : techniquement

- Nodejs
- Mongodb
- Très actif (beaucoup de MAJ : 15 versions depuis le 13 mars)
- API REST

RocketChat : Intégration

- CAS
- SAML
- LDAP
- WebRTC
- Vidéo-conférence (BigBlueButton / Jitsi)

RocketChat : qualités

- C'est hype.
- L'installation semble très facile (avant c'était un peu plus ... sportif)
- L'upgrade est très facile : "rocketchatctl upgrade"
- Backup facile : "rocketchatctl backup", mais les fichiers sont à sauvegarder par une autre méthode.
- Obsolescence programmable des canaux
- Documentation pléthorique.
- Hautement paramétrable.

RocketChat : fonctions "avancées"

- Communication chiffrée de bout en bout (à la demande pour certains chats)
- SMS
- Système de protection contre les "abus de service"
- On peut faire beaucoup de choses avec l'API REST
  - Ajout d'utilisateurs
  - Création de groupes
  - Intégration dans des groupes
  - etc.
- Token API système et token API individuel.

RocketChat : les points d'attention

- Quelques paramétrages "phone home".
- Quelques récupérations de données au dépôt de fichiers (URL).
- Des configurations "théoriques" mais inapplicables.
- Documentation pléthorique, mais on ne trouve pas toujours ce que l'on veut.
- La librairie python (de référence) est totalement insuffisante.
- Concept incompréhensible de l'API REST (Room, canaux, groups, etc.)
- Placer les fichiers en dehors de l'arborescence (destruction des pièces attachées).

Les modifications de la librairie 1

```
pip3 install RocketChatAPI
# Avec BEAUCOUP de modifications derrière, pas forcément monstrueusement compliquées

# dans le fichier api.py
from rocketchat.calls.groups.add_user_room import AddUserRoom
[...]
    def add_user_room(self, user_id, room_id, **kwargs):
        """
        :param user_id:
        :param room_id:
        :param kwargs:
        :return:
        """
        return AddUserRoom(settings=self.settings, **kwargs).call(
            room_id=room_id,
            user_id=user_id,
            **kwargs
        )
```

Les modifications de la librairie 2

```
[root@thor rocketchat]# cat /usr/local/lib/python3.6/site-packages/rocketchat/calls/groups/add_user_room.p
import logging
import json
from rocketchat.calls.base import PostMixin, RocketChatBase
logger = logging.getLogger(__name__)

class AddUserRoom(PostMixin, RocketChatBase):
    endpoint = '/api/v1/groups.invite'

    def build_endpoint(self, **kwargs):
        return self.endpoint

    def build_payload(self, **kwargs):
        return json.dumps({
            'userId': kwargs.get('user_id'),
            'roomId': kwargs.get('room_id')
        })
    def post_response(self, result):
        return result
```

RocketChat : ajout de log dans le chat.

```
uri = 'https://chat.ut-capitole.fr/hooks/SD345fsdzekY4tT7bLDSaN9c/Fdze34a7Drf6a8geouXsxfCMi5DfdsG'
f_log="/var/log/neodev.log"

f = subprocess.Popen(['tail','-F',f_log],\
        stdout=subprocess.PIPE,stderr=subprocess.PIPE)
p = select.poll()
p.register(f.stdout)

while True:
    if p.poll(1):
        data={}
        text=f.stdout.readline().decode("utf-8")
        icon_emoji=':newspaper:'
        if not re.search('(wDrop|unisphere)', text):
            if re.search('^NOTIFY', text):
                icon_emoji=':biohazard:'
            data = {
                "username": "Neodev",
                "icon_emoji": icon_emoji,
                "text": text,
            }
            r = requests.post(uri, json.dumps(data)).content
            print(r)
            time.sleep(1)
```

RocketChat : ajout d'utilisateurs/groupe suivant un fichier

```
with requests.sessions.Session() as session:
    api = RocketChatAPI(settings={'username': username, 'password': password, 'domain': domain},session=se
    l_rooms=api.get_private_rooms()
    s_rooms=set()
    for d_room in l_rooms:
        s_rooms.add(d_room['name'])

    s_users_global=api.get_username_list()

    for user in d_mail.keys():
        if user not in s_users_global:
            mail=d_mail[user]
            nom=d_nom[user]
            temp_password=randomString()
            print("* NOTICE * Création:",user,nom,mail,temp_password)
            api.create_user(mail,nom,temp_password,user)
            s_users_global.add(user)
            time.sleep(10)
    s_users_partage=set()
    files = []
    for r,d,f in os.walk(rep):
       for file in f:
```

Conclusions

- BIEN plus facile maintenant qu'il y a 6 mois. (CAS, création des utilisateurs, etc.).
- C'est rigolo
- C'est sexy
- C'est facile à installer, et à administrer techniquement.
- Fire and Forget...
- Ça ne remplace ni le mail, ni le face à face.
- Ça ne sert presque plus en dehors du confinement.
- C'est compliqué pour administrer techniquement le fonctionnel.
- Sécurité par la périmétrie. Pas sur pour le reste.
- Et pour l'instant, la direction a son Whaller...

Questions ?

Des questions ?