AUTOREN
LOÏS KAUFUNGEN UND MATTIA GALLICCHIO

mammatus - Technische Dokumentation

VIRTUELLE INFRASTRUKTUR - BACKEND

Python Flask Backend

Als Backend haben wir eine REST API in Python Flask geschrieben.

Aufbau

- server.py - der eigentliche server
- vm.py - erstellung einer vm
- smtp.py - versenden der Mails

Server

Der Server Hat verschiedene Endpoints die bei aufruf einen Task auslösen:

```
# packages
import requests
import json
import asyncio
from vm import create_vm # import aus dem VM file
from flask import Flask, request, Response, jsonify
import threading

app = Flask(__name__)

# Endpoint für Status abfrage durch GET oder status ändern duchr POST mit dem befehl - zum Bsp.
"restart"
@app.route('/vm/<vmid>/control', methods=['GET', 'POST'])
def vmControl(vmid):
    if request.method == 'GET':
        url = 'url' + default_client_node + '/qemu/'+ vmid + '/status/current'
        r = requests.get(url, headers=headers, verify=False)
        return r.json()
    elif request.method == "POST":
        data = json.loads(request.data)
        url = 'url' + default_client_node + '/qemu/'+ vmid + '/status/' + data['cmd']
        r = requests.post(url, headers=headers, verify=False)
        return jsonify("success"), r.status_code

# Endpoint für das erstellen einer neuen Virtuellen Maschine
@app.route('/vm/create', methods=['POST'])
def vmCreate():
    data = json.loads(request.data)
    thread = threading.Thread(create_vm(data))
    thread.daemon = True         # Daemonize
    thread.start()
    return jsonify("success")


# App starten
if __name__ == '__main__':
    app.run(host="0.0.0.0", port=int("3000"), debug=True)
```

VM erstellung

```
# packages
import requests
import json
import time
from smtp import smtp_vm_finished, smtp_vm_order # import aus dem SMTP file

# vm erstellung starten
def create_vm(obj):

    #parse json
    obj1 = obj['newvm']
    obj2 = obj['config']

    # send order mail
    smtp_vm_order(obj['user']['mail'], obj['user']['name'])

    # create clone
    url = 'url' + default_client_node + '/qemu/300/clone'
    r = requests.post(url, headers=headers, verify=False, json=obj1)
    task = r.json()
    upid = task['data']
    print("clone started")

    # check status
    while True:
        time.sleep(15)
        url = 'url' + default_client_node + '/tasks/' + str(upid) + '/status'
        r = requests.get(url, headers=headers, verify=False)
        data = r.json()
        if data['data']['status'] == 'stopped' and data['data']['exitstatus'] == 'OK':
            print("clone finished")
            break

    # config vm
    time.sleep(1)
    url = 'url' + default_client_node + '/qemu/'+ obj1['newid'] + '/config'
    r = requests.post(url, headers=headers, verify=False, json=obj2)
    print("vm configured")

    # start vm
    url = 'url + default_client_node + '/qemu/'+ obj1['newid'] + '/status/start'
    r = requests.post(url, headers=headers, verify=False)
    print("vm started")

    # send vm information mail
    smtp_vm_finished(obj['user']['mail'], obj['user']['name'])
```

SMTP

```
import smtplib, ssl
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from email.message import EmailMessage

# VM fertig Bestätigung
def smtp_vm_finished(receiver, name):
    msg = EmailMessage()
    msg['Subject'] = 'Ihre Bestellung ist fertig'
    msg['From'] = 'no-reply@mammatus.ch'
    msg['To'] = receiver

    msg.set_content("mammatus")
    msg.add_alternative( """\
HTML inhatl
    """,  subtype = 'html')

    context = ssl.create_default_context()

    with smtplib.SMTP_SSL("smtp url", port, context=context) as server:

        server.login(user, password)
        server.sendmail(sender, receiver, msg.as_string())



# Bestell bestätigung
def smtp_vm_order(receiver, name):

    msg = EmailMessage()
    msg['Subject'] = 'Bestell Bestätigung'
    msg['From'] = 'no-reply@mammatus.ch'
    msg['To'] = receiver

    msg.set_content("mammatus")
    msg.add_alternative( """\
HTML inhatl
    """,  subtype = 'html')

    context = ssl.create_default_context()

    with smtplib.SMTP_SSL("smtp url", port, context=context) as server:

        server.login(user, password)
        server.sendmail(sender, receiver, msg.as_string())
```