DevOps pour les systèmes, les réseaux et la sécurité

Jean-Marc Pouchoulon

Janvier 2021

1 Mises en situation professionnelle lors du TP et compétences à valider

1.1 Mises en situation professionnelle lors du TP :

- Configurer un environnement de développement Python
- Configurer un ou plusieurs routeurs de marques différentes (Cisco ou Arista) de façon automatisée en utilisant Python ;
- Interroger un annuaire LDAP et consolider les informations ;
- Interroger un DNS et vérifier la cohérence entre un DNS maitre et un DNS esclave ;
- Interroger un calendrier au format ICAL et consolider des information ;
- Scanner un réseau en forgeant des paquets ;
- Créer à l'aide d'un script une instance de machine virtuelle dans le CLOUD AWS ;
- Monitorer le temps de réponse d'une application Web ;
- Générer et manipuler un plan d'adressage par la programmation ;
- Adapter un code existant à l'environnement de l'entreprise ;

1.2 Compétences utilisées ou à acquérir du référentiel de compétences de la licence :

- Maquettez avec VMWARE Workstation, VirtuaBOX et GNS3 ;
- Containeriser un service réseau. (DOCKER/LXC/LXD) ;
- Gérer son infrastructure as code (GIT) ;
- Scripter pour automatiser sa production ou sa sécurité.
- Orchestrer sa production et sa sécurité.
- Superviser sa production et sa sécurité.

Pour le détail des capacités à acquérir et la grille d'évaluation merci de voir le fichier tableur associé sur Moodle.

2 Pré-requis

Avant le démarrage du TP vous devez lire attentivement le notebook suivant que vous pouvez récupérer via :

```
git clone https://registry.iutbeziers.fr:11443/pouchou/tuto1-python-liste-dict.git
```

3 Environnement du TP et aide sur Python

Une image Docker stockée dans le registry de l'IUT est prête à l'emploi pour travailler avec Python3. Pour la première journée, Python 3.8, VS-code installé suffit.

L'éditeur Vim est pré-configuré pour Python (complétion). Des packages clients pour les services réseaux avancés (DNS, LDAP, SSH..) ainsi que pour la gestion des matériels réseaux sont pré-installés.

Pour l'utiliser récupérez l'image Docker :

```
docker network create 3-node_net-0
git config --global http.sslverify false
git clone https://registry.iutbeziers.fr:11443/pouchou/Python-Dev.git
cd Python-Dev
```

Trois environnements de dev Python sont à votre disposition :

— Un léger qui permet juste d'utiliser l'interpréteur python dans sa dernière version avec un fichier source .py dans votre directory de travail sur l'hôte.

```
alias snake="docker run  -v "$(pwd):$(pwd)" -w $(pwd)  python:slim python"
snake test.py
```

— Un environnement lourd (et long à charger même en local), comprenant tous packages python utiles pour le TP et bien plus.

```
docker-compose up -d
docker-compose exec pythondevh bash
ipython # pour lancer l'interpréteur Python 3
```

Le passage en root se fait via la commande

```
sudo su -
```

La commande suivante vous permet d'obtenir directement un prompt ipython dans votre container.

```
docker-compose exec pythondevh  /home/student/.pyenv/shims/ipython
```

Les scripts dans /home/student du container sont sauvegardés dans le volume v-python sur l'hôte. **Si** vous re-instanciez le container avec les commandes suivantes **vous ne perdrez pas** vos scripts et votre notebook :

```
docker-compose down
docker-compose up -d
```

Vim est pré-configuré pour Python dans le container.

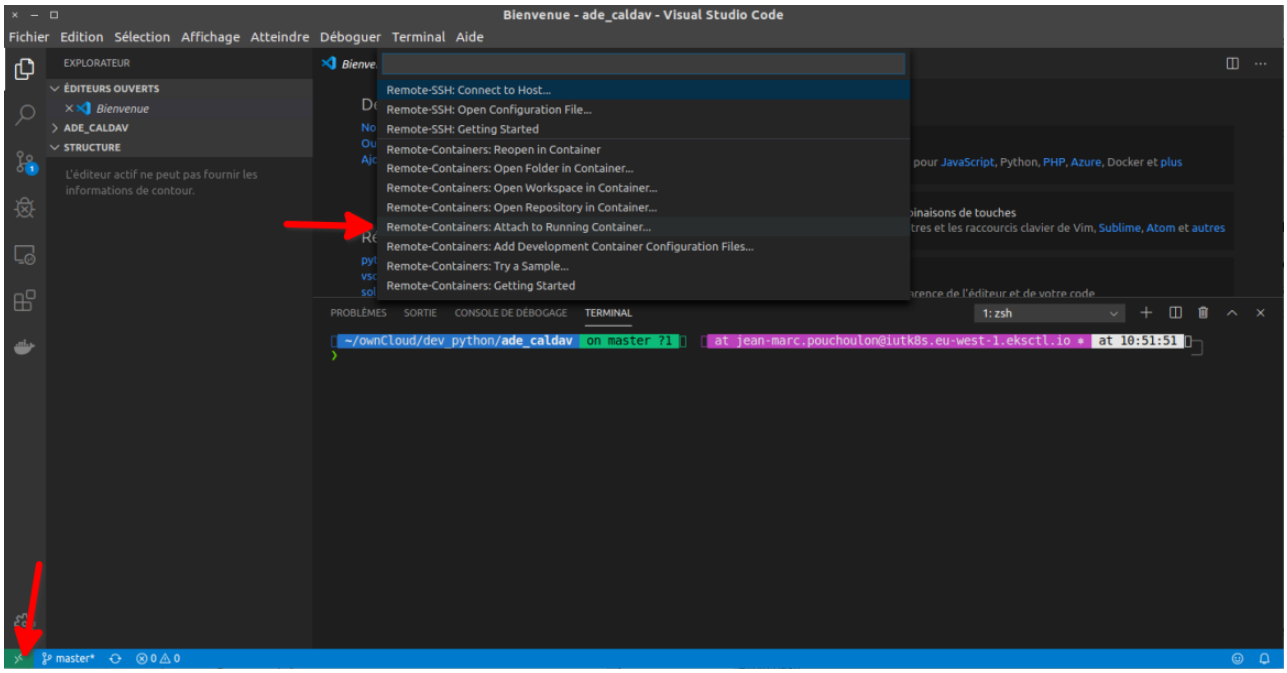
— Un environnement léger qui contient des outils comme jupyter ou ipython mais pas de modules réseaux pré-installés.

```
docker-compose -f ./docker-compose-light.yml up -d
docker-compose -f ./docker-compose-light.yml exec pythondevl  bash
```

Vim c'est bien mais on travaillera surtout avec "Visual Studio Code" qu'il vous ait demandé d'installer [1].

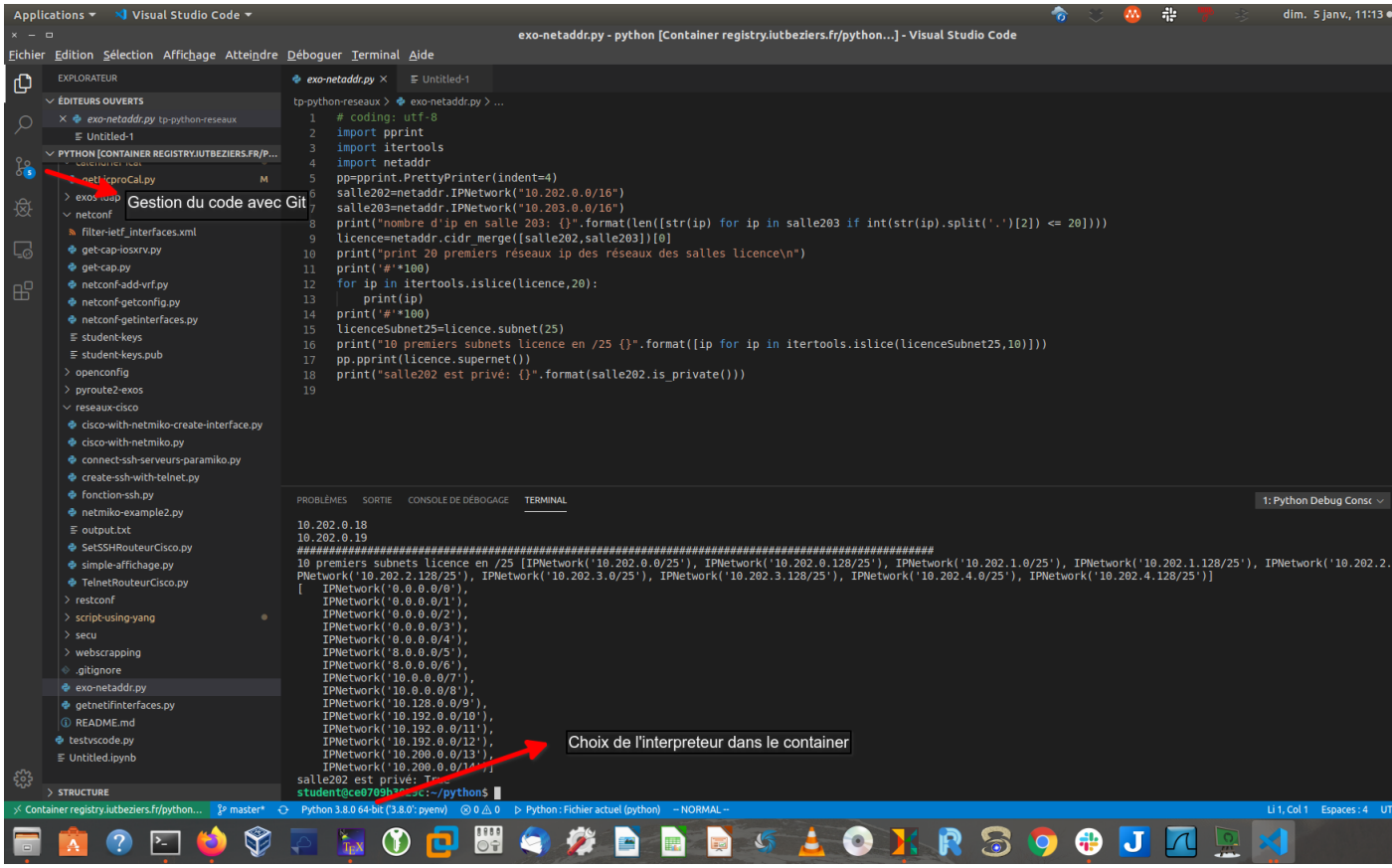
Il faut ensuite installer l'extension Python pour VSCode [2].

Le côté intéressant de vscode est qu'il permet de se connecter en mode "remote" sur un container local [3].
Un fois l'extension installé connectez-vous au container pythondev.



Vous pouvez maintenant utiliser VSCode avec l'environnement Python du container.

1. voir `https://code.visualstudio.com/docs/setup/linux`
2. `https://marketplace.visualstudio.com/items?itemName=ms-python.python`
3. `https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.vscode-remote-extensionpack`



4 Aide sur Python

1. `pypi.org` liste la plupart des modules Python et vous permet souvent d'avoir de la documentation et un lien vers le site d'origine.
2. Vous pouvez avoir l'aide de Python avec la commande help ou dir ou juste utiliser le ? sous ipython :

```
In [18]: import os

In [19]: help(os.chown)

In [1]: import sys

In [2]: sys?
ype:        module
String form: <module 'sys' (built-in)>
Docstring:
This module provides access to some objects used or maintained by the
interpreter and to functions that interact strongly with the interpreter.

Dynamic objects:

argv -- command line arguments; argv[0] is the script pathname if known
path -- module search path; path[0] is the script directory, else ''
modules -- dictionary of loaded modules
....
In [1]: maliste=['jmp','fc','yh']

In [2]: dir(maliste)
```

```
Out[2]:
['__add__',
'__class__',
'__contains__',
'__delattr__',
...
```

3. Obtenir du debug sur le load des packages Python export PYTHONVERBOSE=1

dir(objet) permet de lister les méthodes (propriétés ou fonctions d'un objet).

Pour chaque exercice où un "starter" vous est proposé , utilisez ipython/jupyter pour explorer le contenu des variables et les méthodes des packages. Tous les éléments vous sont donnés il faut "juste vous approprier" le code.

4.1 Outils Python pour le TP

Vous pouvez utiliser ipython afin de mettre au point vos scripts.

```
ipython
%edit monscript.py # Permet d'éditer et d'exécuter un script Python
%run monscript.py # Permet d'exécuter un script. Vous pouvez ensuite examiner les variables
%run -d -b614 monscript.py  # Debug du script , tapez c + run à la ligne 614
```

La fonction breakpoint permet de débugguer le code en créant un ... point d'arrêt.

```
breakpoint()#
```

Vous utiliserez avantageusement Jupyter notebook pour rédiger votre compte-rendu. Jupyter notebook sera accessible depuis votre navigateur sur le port 8888. Il permet aussi de générer une présentation web (au format reveal.js).

Pour le lancer :

```
# remplacez 192.168.16.2 par l'IP de votre container
jupyter notebook --allow-root --no-browser --NotebookApp.token='' --ip=192.168.16.2 --port=8888
```

Il est accessible depuis votre navigateur sur le port 8888. Il permet de rédiger un contenu interactif ou d'accéder à l'interpréteur ipython de l'hôte.
ipython est une version sur-vitaminée de l'interpréteur Python et conserve l'historique des commandes.

5 Interrogation du DNS avec Python

1. Utilisez le module Python socket afin de résoudre www.iutbeziers.fr en adresse IP.
2. Utilisez le module dnspython afin de résoudre www.iutbeziers.fr en adresse IP et reciproquement.
3. Donnez la liste des NS, des MX du domaine iutbeziers.fr.
4. Retrouvez le resolver de votre container.

6 Gestion de l'adressage réseaux

Le package netaddr permet de manipuler des adresses réseaux.

1. Créer deux instances de IPNetwork avec les adresses réseaux des salles 202 et 203.
2. Listez et comptez les adresses ip disponibles dans une salle en fonction du nombre de postes dans la salle (On prendra 20 postes par salle). Vous pouvez avantageusement utiliser une "list comprehension" :

```
[str(ip) for ip in salle203 if int(str(ip).split('.')[2]
```

3. Créez un nouvel objet Network en mergeant les réseaux des deux salles (ne pas limiter le nombre de postes pour le merge.)
4. Listez les subnets /25 de ce nouveau réseau.
5. Listez le supernet des réseaux des salles licence.
6. Verifiez que les réseaux des salles de l'IUT suivent la RFC 1918 (adressage privée).
7. En utilisant le package itertools et la fonction islice limitez le nombre de réseaux affichés à chaque print.

7 Pilotage d'un équipement Cisco avec Netconf, Restconf et Yang

Vous pouvez utilisez le container Python orienté réseaux, ou récupérer une ova `http://store.iutbeziers.fr/cisco/DEVASC_VM.ova` ou en encore installer votre propre environnement.

Lancez l'appliance en mode "tiny" et lancez. Stoppez le démarrage et donner plus de RAM et de CPU puis redémarrez.

Donnez l'accès à votre user "cisco"

```
username cisco privilege 15 password cisco
aaa new-model
aaa authentication login default local
aaa authorization exec default local
```

NB : Le

7.1 Installation de l'OVA (csr1100v)

Installez sous vmware workstation l'ova récupérée à l'adresse suivante : `http://store.iutbeziers.fr/cisco/csr1000v-universalk9.16.09.06.ova` Faites le reste des TP Cisco une fois l'installation réalisée.

- install-the-csr1000v-vm.pdf.
- use-netconf-to-access-an-ios-xe-device-1.pdf.
- use-restconf-to-access-an-ios-xe-device.pdf.
- lab—explore-yang-models.pdf

NB : le "helo" du client n'appelle pas de réponse du routeur.

8 Utilisation de packages réseaux génériques pour piloter des routeurs Cisco

Utilisez le même équipement csr1000v.

8.1 Utilisation du package paramiko pour passer une commande sur un node système ou réseau

1. Utilisez le package paramiko pour afficher le résutat de la commande "show ip interfaces brief" sur votre routeur Cisco.
2. Généralisez votre script afin qu'il se connecte à plusieurs routeurs.

8.2 Utilisation du package netmiko pour gérer un équipement réseau

`https://pynet.twb-tech.com/blog/automation/netmiko.html`

`https://codingnetworker.com/2016/03/automate-ssh-connections-with-netmiko/`

1. Utilisez le package Netmiko pour afficher le résutat de la commande "show ip interface brief" sur votre routeur Cisco.
2. Utilisez le package Netmiko pour configurer un autre réseau sur une interface de loopback pour le routeur Cisco.

9 Scapy un package Python pour tester votre sécurité

Scapy est un outil développé en Python et qui permet de forger et d'interagir avec des paquets réseaux. C'est un outil très intéressant pour un apprenti réseaux et sécurité. Vous allez travailler avec un notebok Jupyter. Il faut être root dans le container (voir le sudo ci-dessous). Vous mettrez votre VM en mode pont si vous en utilisez une.

```
# téléchargement du notebook dans votre home directory
cd python
git clone https://registry.iutbeziers.fr:5443/pouchou/scapy_training.git
cd scapy_training
# Lancez le container ( même pile IP que l'hôte, full acces Network )
sudo jupyter notebook --allow-root --no-browser --NotebookApp.token='' --ip=192.168.16.2 --port=8888
```

Apprenez avec le notebook et répondez aux trois questions à sa fin.

10 Utilisation du package requests avec Python

10.1 Créez un script Python permettant de calculer le temps de login à moodle.didex.fr à l'aide du package requests

Aide sur requests : `http://docs.python-requests.org/en/master/user/quickstart/`.

Vous devez grapher le temps de login à moodle.didex.fr en vous connectant avec votre compte.

1. Complétez le script Python suivant exécutant une boucle qui fera dix mesures espacées d'une seconde. La fonction de récupération du temps est la suivante :

```
# coding: UTF-8
import requests
import timeit
import time

def getTemps():
    """ cette fonction retourne le temps de login à moodle didex.fr"""
    payload={'username':'jean-marc.pouchoulon@iutbeziers.fr','password':'motdepasse'}
    r = requests.post('https://moodle.didex.fr/moodle/login/index.php',data=payload)
    return... ( Retournez ici le temps nécessaire pour faire la requête)
```

2. Graphez via un notebook Jupyter et matplotlib le temps de réponse.

11 Analyse de l'emploi du temps de la licence pro ASUR

1. Récupérez tous les évenements ical de la licence en complétant ce script :

```python
import re
import requests
from ics import Calendar
from collections import Counter

urlbegin='https://proseconsult.umontpellier.fr/jsp/custom/modules/plannings/anonymous_cal.jsp'
urlarg='?code=GRP_BLPMR1_601&projectId=3&calType=ical&firstDate=2020-10-01&lastDate=2021-08-31'
url= urlbegin + urlarg
try:
    c = Calendar(requests.get(url).text)
except UnicodeDecodeError:
    print("pas de données retournées")
    sys.exit()


```

2. Créez une liste de tous les enseignants (Utilisez le package set afin n'avoir qu'une seule occurence de chaque enseignant (trick : list(set(list))).Attention il peut y avoir deux enseignants par event)
   Voilà la structure d'un event :

```python
print(event)
#UID:ADE602d4955542d42657a6965727323031362d323031372d323133302d31382d30
#END:VEVENT
#BEGIN:VEVENT
#CREATED:19700101T000000Z
#LAST-MODIFIED:20160621T143101Z
#SEQUENCE:1617359461
#DTSTAMP:20160621T143100Z
#DTSTART:20170210T080000Z
#DTEND:20170210T090000Z
#SUMMARY:Eng- Anglais (TD)
#DESCRIPTION:\nASUR\nLUNDIN Brigitte\n(Exporté le:21/06/2016 16:31)
#LOCATION:D020
# Cas de deux enseignants
#DESCRIPTION:\nASUR\nPOUCHOULON Jean-Marc\nAZE Jerome\n(Exporté le:19/08/2
```

3. Comptez le nombre d'heures réalisées par enseignant.

12 Gestion des réseaux Linux

12.1 Netifaces

1. Gérer ses interfaces avec Netifaces
   Créez une fonction qui utilise le package Netifaces afin de lister toutes les adresses IPv4 d'une machine.

12.2 Gérer ses réseaux avec pyroute2

Vous pouvez regarder la présentation de Florent Fourcot `https://www.youtube.com/watch?v=_jMryRsi3GM` à la pyconfr 2017 sur pyroute2. Il indique en amont de sa présentation :

Le noyau Linux a des fonctionnalités très avancées pour construire des équipements réseaux. Si des outils existent en ligne de commande pour tout faire (avec les outils iproute2, comme les commandes ip, tc, ss, ...), on est très vite limité si on tente de lancer ces" commandes avec os.system ou équivalent, et d'analyser la sortie des commandes pour lire l'état du système.

Heureusement, les outils iproute2 utilisent en réalité une API pour discuter avec le noyau : netlink. Cette interface est construite sur une simple socket, qui permet de recevoir des informations et d'en écrire. À

travers cette socket, tout est configurable et il devient très simple de construire des applications complexes (surveillance des évènements en écoutant les messages, etc).

Si la plupart des outils utilisant l'interface netlink sont écrits en C (ils n'ont qu'à reprendre les bibliothèques existantes pour iproute2), les développeurs Python ont une bibliothèque complète à disposition : pyroute2 (https ://github.com/svinota/pyroute2).

La documentation de pyroute2 est accessible ici `https://docs.pyroute2.org/iproute.html`

1. Listez les interfaces et les adresses IP de votre container Python avec pyroute2.
2. Créez une interface bridge nommée brctx avec pyroute2 et affectez-lui une adresse mac.

13 Interrogation d'un annuaire LDAP

13.1 Interrogation de l'annuaire LDAP de l'IUT de Béziers (si présentiel)

Vous utiliserez le package python-ldap afin de vous connecter à l'annuaire de l'IUT sur le serveur "scribe.iutbeziers.fr". Servez-vous des éléments de connexion suivants :

- server = 'scribe.iutbeziers.fr'
- baseDN ="ou=utilisateurs,ou=0341884N,ou=ac-montpellier,ou=education,o=gouv,c=fr"
- binddn = "cn=authldap,o=gouv,c=fr"
- passwd = "authldap"

1. Récupérez votre fiche LDAP
2. Comptez le nombre de fiches dans l'annuaire par type de personnel (enseignants et administratifs d'un côté et étudiants de l'autre)
   Vous pouvez utilisez Counter du package Collections et le tips suivant pour "mettre à plat la liste de listes retournée par python-ldap" :

```
>>> import operator
>>> l = [[1,2,3],[4,5,6], [7], [8,9]]
>>> reduce(operator.concat, l)
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

13.2 Interrogation de l'annuaire LDAP de test (si distanciel)

Si le TP est en distanciel vous pouvez installer un annuaire de test de la façon suivante :

```
LDAP_CID=$(docker run -d --rm  -p 389:389 -p 636:636 --hostname ldap.iutbeziers.fr   registry.iutbeziers.fr/ldapiut:demo --copy-service --logleve
sleep 5
# alimentation de l'annuaire et création des schémas
docker exec $LDAP_CID bash -c '/root/iutbeziers.sh'
# Liste des entrées
docker exec $LDAP_CID bash -c  'ldapsearch -x -H ldap://localhost -b dc=gouv,dc=fr -s sub -D "cn=admin,dc=gouv,dc=fr" -w iutbrt'
#Comme le container écoute sur la machine vous pouvez passer des requêtes ldap sans être dans le container (attention ne pas lancer un slapd su
ldapsearch -x -H ldap://localhost -b dc=gouv,dc=fr -s sub -D "cn=admin,dc=gouv,dc=fr" -w
iutbrt
# purger
docker stop $(docker ps|grep ldapiut|awk '{print $1}') && docker rm $(docker ps|grep ldapiut|awk '{print $1}')
```

En transposant en Python répondez aux questions suivantes :

1. Retrouvez les 16 OU obtenues avec la commande suivante.

   ```
   ldapsearch  -x -w iutbrt  "(objectClass=organizationalUnit)"  dn -LLL|grep dn
   ```

2. Retrouvez le DN suivant

```
ldapsearch  -LLL -x -w iutbrt -s base  -b "cn=CamileMarples,ou=Administratif,dc=gouv,dc=fr"
```

3. Combien de personnes appartiennent à l'OU Reseaux ?

```
ldapsearch  -x -w iutbrt  "(&(objectClass=inetOrgPerson)(ou=Reseaux))"
```

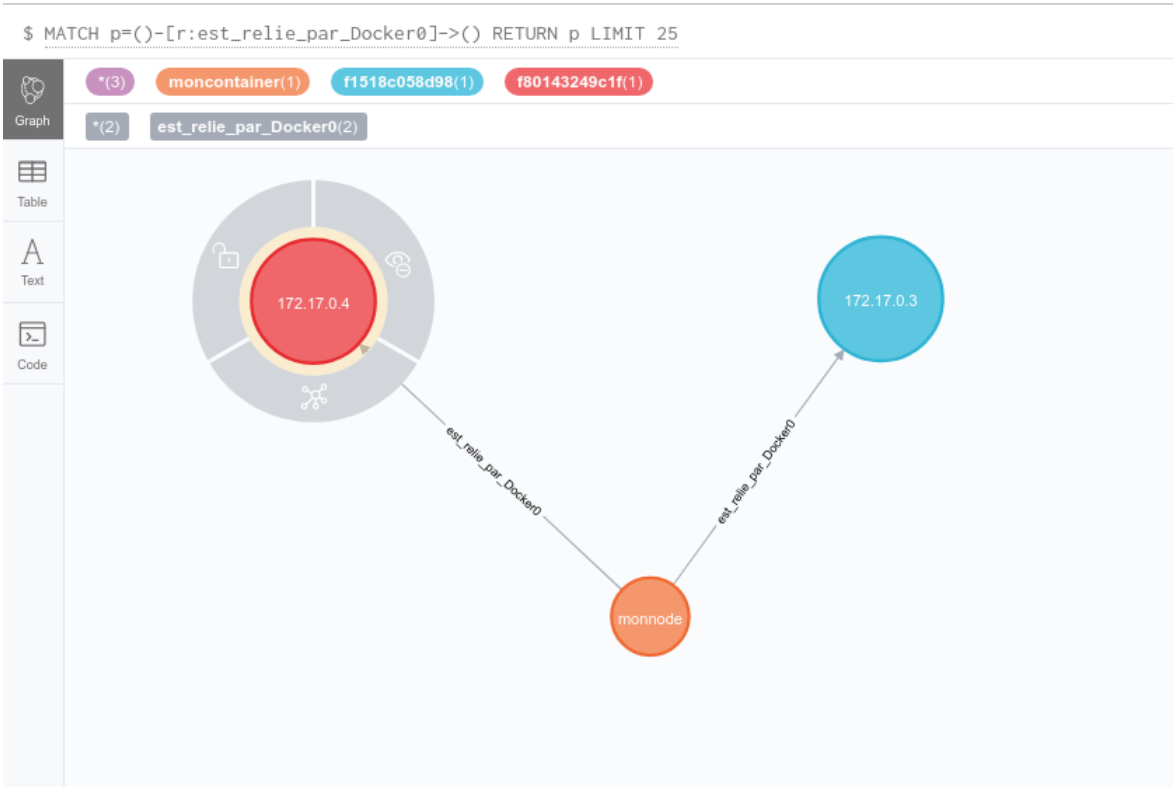
14 Gérer des machines virtuelles dans le CLOUD AWS (Amazon Web Services)

En utilisant votre compte Rosetta (`https://www.rosettahub.com/welcome`), récupérez la liste de vos VM en utilisant le package python boto3.

```
from zeep import Client
import boto3


client=Client('https://api.rosettahub.com/public/PlatformServices?wsdl')
platform = client.service
userLogin = "jean-marc.pouchoulon"
userPwd = "monpass"


timeOut = 0
session = platform.newSession(userLogin, userPwd,timeOut)

users = platform.cpocGetFederatedUsers( session, {'organizationName':'UMONTPELLIER-IUT Béziers'})
```

14.1 Tracer le voisinage reseau en récuperant des informations via protocole LLDP

1. Le protocole lldp ("Link Layer Discovery Protocol") est un équivalent du protocole propriétaire CDP ("Cisco Discovery Protocol"). Le nut de cet exercice est de récupérer des informations sur des containers Docker annonçant des informations via lldp et de les publier sur une base orientée graphe (`https://neo4j.com/` qui permet de visualiser des nœuds reliés entre eux par des relations (ici les containers rattachés au bridge Docker0) :



Vous installerez 2 containers Docker avec un daemon snmp et lldpd actif :

```
docker run  --rm   -d --privileged -v /sys/fs/cgroup:/sys/fs/cgroup:ro registry.iutbeziers.fr/debian:bustersyd

      sudo lldpctl  -f json
      {
            "lldp": {
                  "interface": [
                  {
                        "eth0": {
                              "via": "CDPv2",
                              "rid": "1",
                              "age": "0 day, 00:00:31",
                              "chassis": {
                                    "21dfebd19197": {
                                          "id": {
                                                "type": "local",
                                                "value": "21dfebd19197"
                                          },
                                          "descr": "Linux running on\nDebian GNU/Linux 10 (buster) Linux 5.0.21-050021-generic #2019
                                          "mgmt-ip": "172.17.0.2",
                                          "capability": {
                                                "type": "Router",
                                                "enabled": true
                                          }
                                    }
                              },
                              "port": {
                                    "id": {
                                          "type": "ifname",
                                          "value": "eth0"
                                    },
                                    "descr": "eth0",
                                    "ttl": "120"
```

```
                        }
                    }
                },
                {
                    "eth0": {
                        "via": "CDPv2",
                        "rid": "2",
                        "age": "0 day, 00:00:03",
                        "chassis": {
                            "a98d4e9d8500": {
                                "id": {
                                    "type": "local",
                                    "value": "a98d4e9d8500"
                                },
                                "descr": "Linux running on\nDebian GNU/Linux 10 (buster) Linux 5.0.21-050021-generic #2019
                                "mgmt-ip": "172.17.0.3",
                                "capability": {
                                    "type": "Router",
                                    "enabled": true
                                }
                            }
                        },
                        "port": {
                            "id": {
                                "type": "ifname",
                                "value": "eth0"
                            },
                            "descr": "eth0",
                            "ttl": "120"
                        }
                    }
                }
                ]
        }
    }
```


L'installation d'un container neo4j vous permettra de visualiser les liens entre les containers :

```
docker run \
--name testneo4j \
-p7474:7474 -p7687:7687 \
-d \
-v $HOME/neo4j/data:/data \
-v $HOME/neo4j/logs:/logs \
-v $HOME/neo4j/import:/var/lib/neo4j/import \
-v $HOME/neo4j/plugins:/plugins \
--env NEO4J_AUTH=neo4j/test
neo4j:latest
```

Vous utiliserez avantageusement les packages :

- subprocess pour lancez la commmande lldpctl dans votre code Python.
- simplejson pour transformer le texte reçu de subprocess en json
- glom pour parser json et récupérer les informations chassis ip description issues de la sortie de la commande lancée par subprocess.
- py2neo pour travailler stocker des informations dans neo4j.

L'utilisation d'une dataclass (à partir de Python 3.7) est intéressante pour créer des objets simples. Voilà le début du code :

```
# coding: utf-8
import subprocess
import simplejson as json
from glom import glom
from dataclasses import dataclass
from py2neo import Graph, Node, Relationship

# Création d'une dataclasse qui permettra ensuite de créer les nodes sur neo4j:

@dataclass
class NetworkNode:
        name : str
        ipaddress: str
        descr: str



graph = Graph("bolt://IP_Hôte:7687",user='neo4j',password='pass')
graph.delete_all()

out=json.loads(subprocess.check_output("sudo lldpctl -f json",shell=True).decode('UTF-8'))
interfaces=glom(out,'lldp.interface')

list_of_nodes = list()
```