Python fosstorrent tool

Things it does:

1) Automate setup of docker containers
2) Get all .torrent files from xml url

Doesnt do:

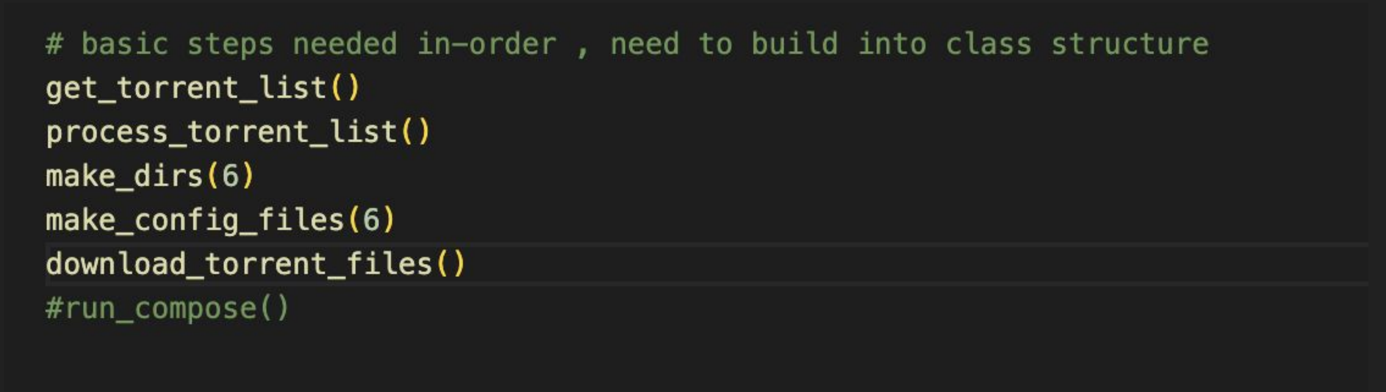
1) Automatic add new torrents
2) Show status of torrents
3) manage stuck or failed services

Also applicable to qbittorrent - need to modify docker example, same watch-dir config applies

Basic functions outline

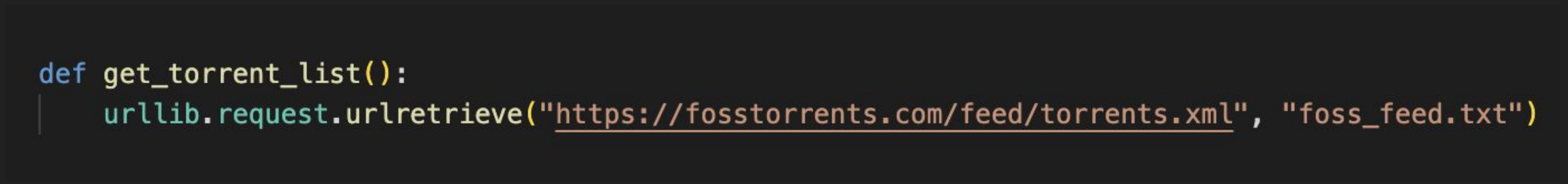
Main.py has 6 functions defined

- Get torrent list
- Process torrent list
- Make dirs
- Make config files
- Download torrent files
- Run compose



```
# basic steps needed in-order , need to build into class structure
get_torrent_list()
process_torrent_list()
make_dirs(6)
make_config_files(6)
download_torrent_files()
#run_compose()
```

get_torrent_list



```python
def get_torrent_list():
    urllib.request.urlretrieve("https://fosstorrents.com/feed/torrents.xml", "foss_feed.txt")
```

Thats all, download /feed/torrents.xml into the current dir (where the python script ran) , save it as “foss_feed.txt”

process_torrent_list

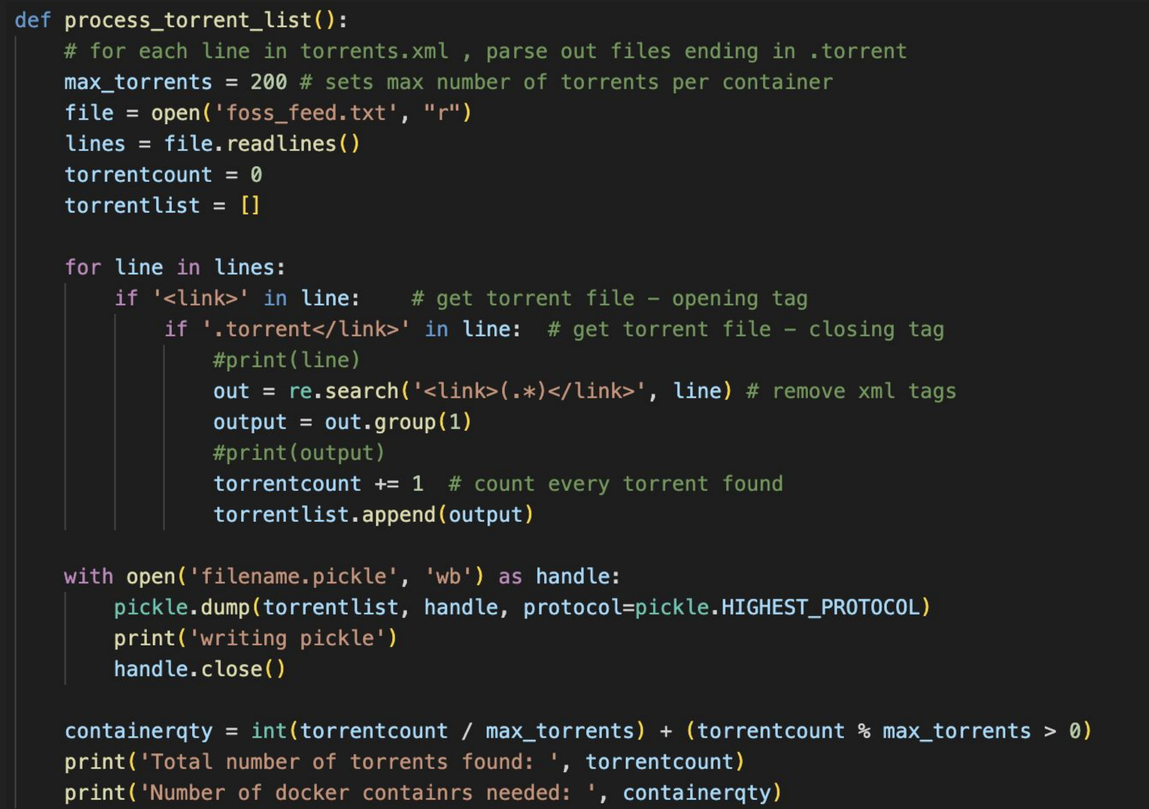
How it works:

Open foss_feed.txt
For each line in this file, do the following

1) Check if the line starts with <link>
2) If it starts with link, check if it ends with </link>
3) If we have the above so far, then save the text between <link>example.torrent</link> so that we get “example.torrent”
4) Keep working through each line, and add the .torrent names to a list

Once the list of torrents is parsed, do the following:

1) Save the python list as a pickle , so we can re-read it later without parsing it again

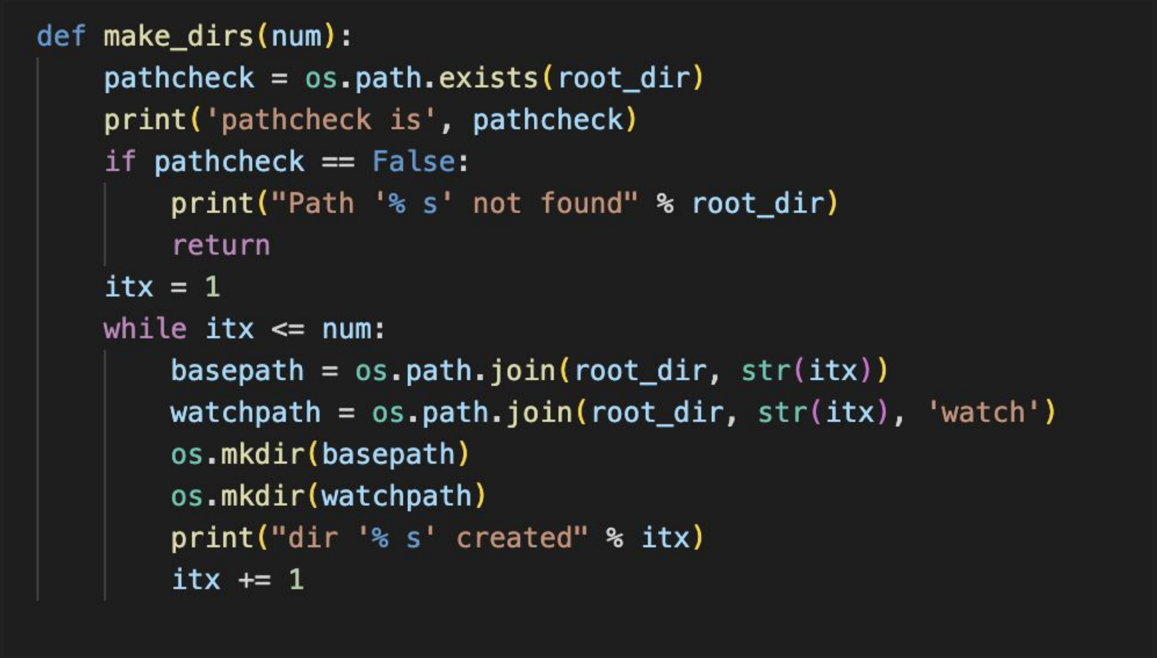


```python
def process_torrent_list():
    # for each line in torrents.xml , parse out files ending in .torrent
    max_torrents = 200 # sets max number of torrents per container
    file = open('foss_feed.txt', "r")
    lines = file.readlines()
    torrentcount = 0
    torrentlist = []

    for line in lines:
        if '<link>' in line:    # get torrent file - opening tag
            if '.torrent</link>' in line:  # get torrent file - closing tag
                #print(line)
                out = re.search('<link>(.*)</link>', line) # remove xml tags
                output = out.group(1)
                #print(output)
                torrentcount += 1  # count every torrent found
                torrentlist.append(output)

    with open('filename.pickle', 'wb') as handle:
        pickle.dump(torrentlist, handle, protocol=pickle.HIGHEST_PROTOCOL)
        print('writing pickle')
        handle.close()

    containerqty = int(torrentcount / max_torrents) + (torrentcount % max_torrents > 0)
    print('Total number of torrents found: ', torrentcount)
    print('Number of docker containrs needed: ', containerqty)
```

make_dirs(num)

Example output:

```
east:/fourdiskpool/fosstorrent_root$ python3 main.py
writing pickle
Total number of torrents found:  1183
Number of docker containrs needed:  6
pathcheck is True
dir '1' created
dir '2' created
dir '3' created
dir '4' created
dir '5' created
dir '6' created
```



```python
def make_dirs(num):
    pathcheck = os.path.exists(root_dir)
    print('pathcheck is', pathcheck)
    if pathcheck == False:
        print("Path '% s' not found" % root_dir)
        return
    itx = 1
    while itx <= num:
        basepath = os.path.join(root_dir, str(itx))
        watchpath = os.path.join(root_dir, str(itx), 'watch')
        os.mkdir(basepath)
        os.mkdir(watchpath)
        print("dir '% s' created" % itx)
        itx += 1
```

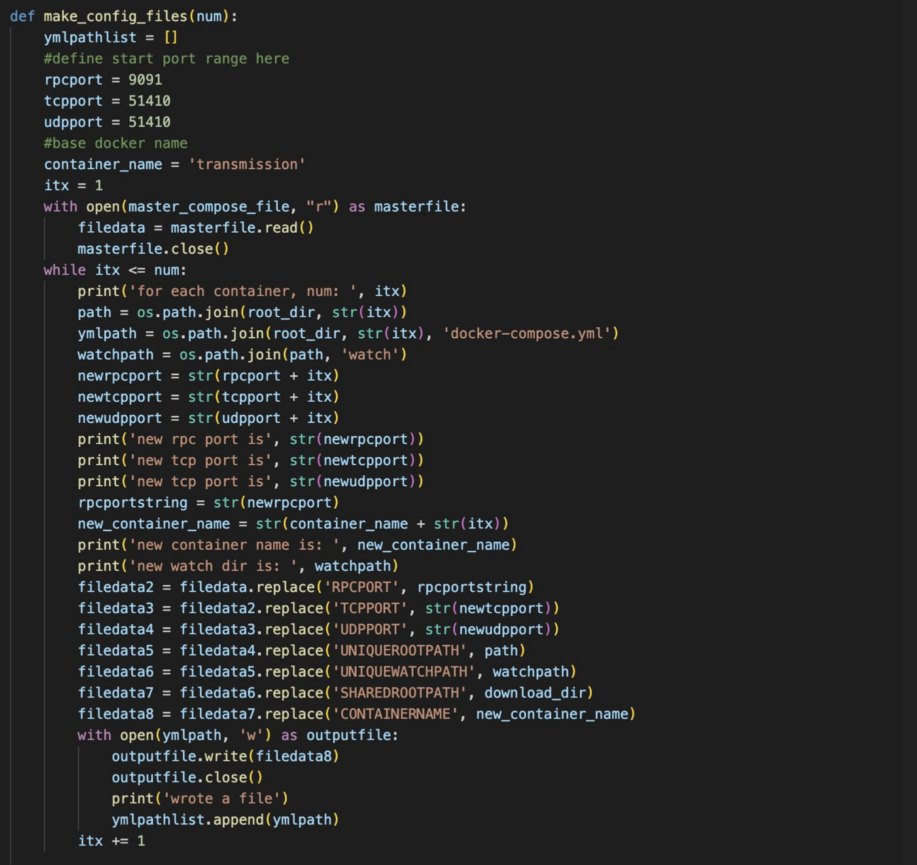
How it works:

1) Confirm the root path exists
2) Create x many dirs

make_config_files(num)

This is a workaround for docker to set unique watch-paths for each container

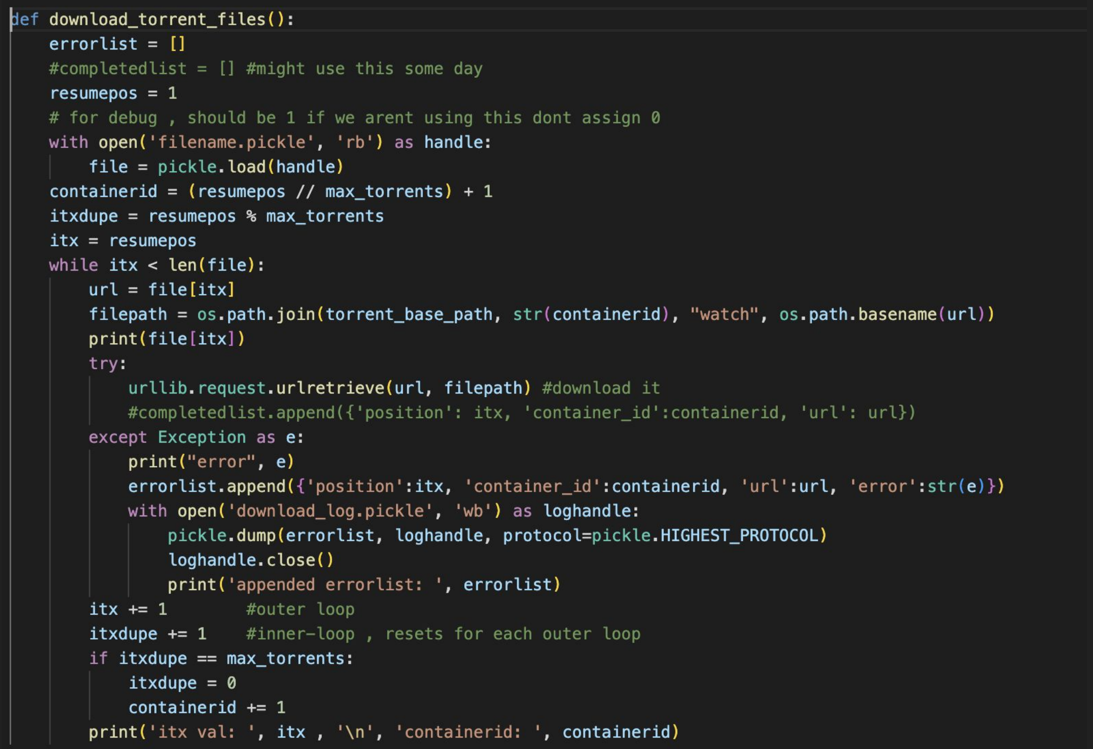
```
$ cat docker-compose-example.txt
---
version: "2.1"
services:
  transmission:
        image: lscr.io/linuxserver/transmission:latest
        container_name: CONTAINERNAME
        environment:
        - PUID=1000
        - PGID=1000
        - TZ=Europe/London
        - USER=username #optional
        - PASS=password #optional
        - WHITELIST=127.0.0.1 #optional
        volumes:
        - UNIQUEROOTPATH:/config # only seen by this container
        - UNIQUEWATCHPATH:/watch # only seen by this container
        - SHAREDROOTPATH:/downloads # same on all containers
        ports:
        - RPCPORT:RPCPORT
        - TCPPORT:TCPPORT
        - UDPPORT:UDPPORT/udp
        restart: unless-stopped
```



```python
def make_config_files(num):
    ymlpathlist = []
    #define start port range here
    rpcport = 9091
    tcpport = 51410
    udpport = 51410
    #base docker name
    container_name = 'transmission'
    itx = 1
    with open(master_compose_file, "r") as masterfile:
        filedata = masterfile.read()
        masterfile.close()
    while itx <= num:
        print('for each container, num: ', itx)
        path = os.path.join(root_dir, str(itx))
        ymlpath = os.path.join(root_dir, str(itx), 'docker-compose.yml')
        watchpath = os.path.join(path, 'watch')
        newrpcport = str(rpcport + itx)
        newtcpport = str(tcpport + itx)
        newudpport = str(udpport + itx)
        print('new rpc port is', str(newrpcport))
        print('new tcp port is', str(newtcpport))
        print('new tcp port is', str(newudpport))
        rpcportstring = str(newrpcport)
        new_container_name = str(container_name + str(itx))
        print('new container name is: ', new_container_name)
        print('new watch dir is: ', watchpath)
        filedata2 = filedata.replace('RPCPORT', rpcportstring)
        filedata3 = filedata2.replace('TCPPORT', str(newtcpport))
        filedata4 = filedata3.replace('UDPPORT', str(newudpport))
        filedata5 = filedata4.replace('UNIQUEROOTPATH', path)
        filedata6 = filedata5.replace('UNIQUEWATCHPATH', watchpath)
        filedata7 = filedata6.replace('SHAREDROOTPATH', download_dir)
        filedata8 = filedata7.replace('CONTAINERNAME', new_container_name)
        with open(ymlpath, 'w') as outputfile:
            outputfile.write(filedata8)
            outputfile.close()
            print('wrote a file')
            ymlpathlist.append(ymlpath)
        itx += 1
```

Function: Download torrent files

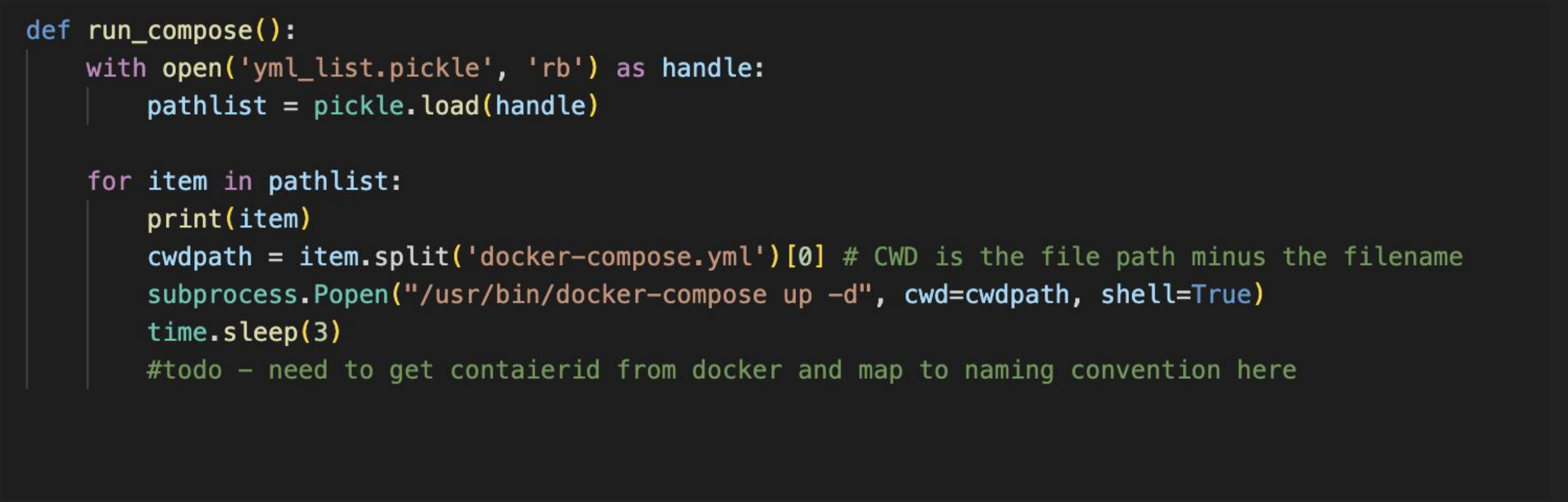
```
east:/fourdiskpool/fosstorrent_root$ python3 main.py
https://fosstorrents.com/files/0ad-0.0.26-alpha-osx64.dmg.torrent
itx val:  2
 containerid:  1
https://fosstorrents.com/files/0ad-0.0.26-alpha-win32.exe.torrent
itx val:  3
 containerid:  1
https://fosstorrents.com/files/0ad-0.0.26-alpha-unix-build.tar.gz.torrent
itx val:  4
 containerid:  1
https://fosstorrents.com/files/0ad-0.0.26-alpha-unix-build.tar.xz.torrent
itx val:  5
 containerid:  1
https://fosstorrents.com/files/0ad-0.0.26-alpha-unix-data.tar.gz.torrent
itx val:  6
 containerid:  1
https://fosstorrents.com/files/0ad-0.0.26-alpha-unix-data.tar.xz.torrent
itx val:  7
 containerid:  1
https://fosstorrents.com/files/absolute64-20220825.iso.torrent
itx val:  8
 containerid:  1
https://fosstorrents.com/files/absolute64-live-current.iso.torrent
itx val:  9
 containerid:  1
```



```python
def download_torrent_files():
    errorlist = []
    #completedlist = [] #might use this some day
    resumepos = 1
    # for debug , should be 1 if we arent using this dont assign 0
    with open('filename.pickle', 'rb') as handle:
        file = pickle.load(handle)
    containerid = (resumepos // max_torrents) + 1
    itxdupe = resumepos % max_torrents
    itx = resumepos
    while itx < len(file):
        url = file[itx]
        filepath = os.path.join(torrent_base_path, str(containerid), "watch", os.path.basename(url))
        print(file[itx])
        try:
            urllib.request.urlretrieve(url, filepath) #download it
            #completedlist.append({'position': itx, 'container_id':containerid, 'url': url})
        except Exception as e:
            print("error", e)
            errorlist.append({'position':itx, 'container_id':containerid, 'url':url, 'error':str(e)})
            with open('download_log.pickle', 'wb') as loghandle:
                pickle.dump(errorlist, loghandle, protocol=pickle.HIGHEST_PROTOCOL)
                loghandle.close()
                print('appended errorlist: ', errorlist)
        itx += 1        #outer loop
        itxdupe += 1    #inner-loop , resets for each outer loop
        if itxdupe == max_torrents:
            itxdupe = 0
            containerid += 1
        print('itx val: ', itx , '\n', 'containerid: ', containerid)
```

run_compose

Function to run docker-copose in each of the new dirs, scales up to however many num of dirs were created in the earlier step



```python
def run_compose():
    with open('yml_list.pickle', 'rb') as handle:
        pathlist = pickle.load(handle)

    for item in pathlist:
        print(item)
        cwdpath = item.split('docker-compose.yml')[0] # CWD is the file path minus the filename
        subprocess.Popen("/usr/bin/docker-compose up -d", cwd=cwdpath, shell=True)
        time.sleep(3)
        #todo - need to get contaierid from docker and map to naming convention here
```