Configurez une petite image alpine de docker de Linux sur IOx

Contenu

Introduction

Ce document décrit le processus de configuration pour créer, déployer et gérer des applications basées sur docker sur les périphériques IOx-capables de Cisco.

Conditions préalables

Conditions requises

Aucune spécification déterminée n'est requise pour ce document.

[Composants utilisés](#)

Les informations contenues dans ce document sont basées sur les versions de matériel et de logiciel suivantes :

- Périphérique capable d'IOx qui est configuré pour IOx :
  Adresse IP configuréeSystème d'exploitation client (gos) et s'exécuter de cadre d'applications de Cisco (CAF)Traduction d'adresses de réseau (NAT) configuré pour l'accès à CAF (port 8443)NAT configuré pour l'accès au shell gos (port 2222)
- Hôte de Linux (une installation minimale de CentOS 7 est utilisés pour cet article)
- Fichiers d'installation de client d'IOx dont peut être téléchargé : https://software.cisco.com/download/release.html?mdfid=286306005&softwareid=286306762

Les informations contenues dans ce document ont été créées à partir des périphériques d'un environnement de laboratoire spécifique. Tous les périphériques utilisés dans ce document ont démarré avec une configuration effacée (par défaut). Si votre réseau est opérationnel, assurez-vous que vous comprenez l'effet potentiel de toute commande.

[Informations générales](#)

IOx peut héberger différents types de Javas de modules principalement, de python, de LXC, de virtual machine (VM) etc., et il peut également exécuter des conteneurs de docker. Cisco offre une image de base et un référentiel complet de hub de docker : https://devhub.cisco.com/artifactory/webapp/#/artifacts/browse/tree/General/iox-docker qui peut être utilisé pour construire des conteneurs de docker.

Voici un guide pas à pas sur la façon dont construire un conteneur simple de docker avec l'utilisation du Linux alpin. Le Linux alpin est une petite image de Linux (autour de 5MB), qui est employée souvent comme base pour des conteneurs de docker. En cet article, vous commencez à partir d'un périphérique configuré d'IOx, une machine Linux et vous vides de CentOS 7 établissent un petit web server de python, l'empaquettent dans un conteneur de docker et déploient cela sur un périphérique d'IOx.

Configurez

1. Installez et préparez le client d'IOx sur l'hôte de Linux.

Le client d'IOx est l'outil qui peut empaqueter des applications et communiquer avec le périphérique IOx-capable pour gérer des applications d'IOx.

Après que vous téléchargiez le module d'installation ioxclient, il peut être installé comme suit :

```
[jedepuyd@db ~]$ ll ioxclient_1.3.0.0_linux_amd64.tar.gz
-rw-r--r--. 1 jedepuyd jedepuyd 4668259 Jun 22 09:19 ioxclient_1.3.0.0_linux_amd64.tar.gz

[jedepuyd@db ~]$ tar -xvzf ioxclient_1.3.0.0_linux_amd64.tar.gz
ioxclient_1.3.0.0_linux_amd64/ioxclient
ioxclient_1.3.0.0_linux_amd64/README.md

[jedepuyd@db ~]$ ./ioxclient_1.3.0.0_linux_amd64/ioxclient --version
Config file not found : /home/jedepuyd/.ioxclientcfg.yaml
Creating one time configuration..
Your / your organization's name : Cisco
Your / your organization's URL : www.cisco.com
Your IOx platform's IP address[127.0.0.1] : 10.48.43.197
Your IOx platform's port number[8443] :
Authorized user name[root] : admin
Password for admin :
Local repository path on IOx platform[/software/downloads]:
URL Scheme (http/https) [https]:
API Prefix[/iox/api/v2/hosting/]:
Your IOx platform's SSH Port[2222]:
Activating Profile default
Saving current configuration
ioxclient version 1.3.0.0

[jedepuyd@db ~]$ ./ioxclient_1.3.0.0_linux_amd64/ioxclient --version
ioxclient version 1.3.0.0
```

Comme vous pouvez voir, sur le premier lancement du client d'IOx, un profil pouvez être généré pour le périphérique d'IOx que vous pouvez gérer avec le client d'IOx. Au cas où vous voudriez faire ceci plus tard ou si vous voulez pour ajouter/modification les configurations, vous pouvez exécuter cette commande plus tard : **les profils ioxclient créent**

2. Installez et préparez le docker sur l'hôte de Linux.

Le docker est utilisé pour construire un conteneur et pour tester l'exécution de notre exemple

d'applications.

Les étapes d'installation pour installer le docker dépend largement du système d'exploitation Linux que vous l'installez en fonction. Pour cet article, vous pouvez utiliser CentOS 7. Pour des instructions d'installation pour différentes distributions, référez-vous : https://docs.docker.com/engine/installation/.

Installez les conditions préalables :

```
[jedepuyd@db ~]$ sudo yum install -y yum-utils device-mapper-persistent-data lvm2
...
Complete!
```

Ajoutez le repo de docker :

```
[jedepuyd@db ~]$ sudo yum-config-manager --add-repo
https://download.docker.com/linux/centos/docker-ce.repo
Loaded plugins: fastestmirror
adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
grabbing file https://download.docker.com/linux/centos/docker-ce.repo to
/etc/yum.repos.d/docker-ce.repo
repo saved to /etc/yum.repos.d/docker-ce.repo
```

Installez le docker (recevez la vérification de clé GPG quand vous installez) :

```
[jedepuyd@db ~]$ sudo yum install docker-ce
...
Complete!
```

Docker de début :

```
[jedepuyd@db ~]$ sudo systemctl start docker[jedepuyd@db iox_docker_pythonweb]$ vi Dockerfile
[jedepuyd@db iox_docker_pythonweb]$ cat Dockerfile
FROM alpine:3.3

RUN apk add --no-cache python
COPY webserver.py /webserver.py
```

Afin de pouvoir accéder à/exécutez le docker en tant qu'utilisateur régulier, ajoutez cet utilisateur au groupe de docker et régénérez l'adhésion à des associations :

```
[jedepuyd@db ~]$ sudo usermod -a -G docker jedepuyd
[jedepuyd@db ~]$ newgrp docker
```

Procédure de connexion au hub de docker :

Le hub de docker contient l'image de base alpine que vous pouvez utiliser. Au cas où vous n'auriez pas un ID de docker encore, vous devez s'enregistrer en fonction : https://hub.docker.com/.

```
[jedepuyd@db ~]$ docker login
Log in with your Docker ID to push and pull images from Docker Hub. If you do not have a Docker
ID, head over to https://hub.docker.com to create one.
Username: jensdepuydt
Password:
Login Succeeded
```

3. Créez le web server de python.

Maintenant que la préparation est faite, vous pouvez commencer à établir l'application réelle qui peut fonctionner sur le périphérique d'IOx-enable.

```
[jedepuyd@db ~]$ mkdir iox_docker_pythonweb
```

```
[jedepuyd@db ~]$ cd iox_docker_pythonweb/
[jedepuyd@db iox_docker_pythonweb]$ vi webserver.py
[jedepuyd@db iox_docker_pythonweb]$ cat webserver.py
#!/usr/bin/env python
from BaseHTTPServer import BaseHTTPRequestHandler, HTTPServer
import SocketServer
import os

class S(BaseHTTPRequestHandler):
    def _set_headers(self):
        self.send_response(200)
        self.send_header('Content-type', 'text/html')
        self.end_headers()

    def do_GET(self):
        self._set_headers()
        self.wfile.write("<html><body><h1>IOX python webserver</h1></body></html>")

def run(server_class=HTTPServer, handler_class=S, port=80):
    server_address = ('', port)
    httpd = server_class(server_address, handler_class)
    print 'Starting webserver...'
    log_file_dir = os.getenv("CAF_APP_LOG_DIR", "/tmp")
    log_file_path = os.path.join(log_file_dir, "webserver.log")
    logf = open(log_file_path, 'w')
    logf.write('Starting webserver....\n')
    logf.close()

    httpd.serve_forever()

if __name__ == "__main__":
    from sys import argv

    if len(argv) == 2:
        run(port=int(argv[1]))
    else:
        run()
```

Ce code est un web server très minimal de python, que vous créez dans webserver.py. Le web server renvoie simplement le web server de python d'IOx dès que GET sera demandé. Le port sur lequel le web server commence peut être le port 80 ou le premier argument donné à webserver.py.

Ce code contient également, dans la fonction de passage, une inscription à un fichier journal. Le fichier journal est disponible pour la consultation du client d'IOx ou du gestionnaire local.

4. Créez le conteneur de Dockerfile et de docker.

Maintenant que vous avez l'application (webserver.py) qui devrait fonctionner dans votre conteneur, il est temps de construire le conteneur de docker. Un conteneur est défini dans un Dockerfile :

```
[jedepuyd@db iox_docker_pythonweb]$ vi Dockerfile
[jedepuyd@db iox_docker_pythonweb]$ cat Dockerfile
FROM alpine:3.3

RUN apk add --no-cache python
COPY webserver.py /webserver.py
```

Comme vous pouvez voir, le Dockerfile est également maintenu simple. Vous commencez par l'image de base alpine, installez le python et copiez votre webserver.py sur la racine du conteneur.

Une fois que vous avez votre Dockerfile prêt, vous pouvez construire le conteneur de docker :

```
jedepuyd@db iox_docker_pythonweb]$ docker build -t ioxpythonweb:1.0 .
Sending build context to Docker daemon 3.584 kB
Step 1/3 : FROM alpine:3.3
3.3: Pulling from library/alpine
10462c29356c: Pull complete
Digest: sha256:9825fd1a7e8d5feb52a2f7b40c9c4653d477b797f9ddc05b9c2bc043016d4819
Status: Downloaded newer image for alpine:3.3
 ---> 461b3f7c318a
Step 2/3 : RUN apk add --no-cache python
 ---> Running in b057a8183250
fetch http://dl-cdn.alpinelinux.org/alpine/v3.3/main/x86_64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.3/community/x86_64/APKINDEX.tar.gz
(1/10) Installing libbz2 (1.0.6-r4)
(2/10) Installing expat (2.1.1-r1)
(3/10) Installing libffi (3.2.1-r2)
(4/10) Installing gdbm (1.11-r1)
(5/10) Installing ncurses-terminfo-base (6.0-r6)
(6/10) Installing ncurses-terminfo (6.0-r6)
(7/10) Installing ncurses-libs (6.0-r6)
(8/10) Installing readline (6.3.008-r4)
(9/10) Installing sqlite-libs (3.9.2-r0)
(10/10) Installing python (2.7.12-r0)
Executing busybox-1.24.2-r1.trigger
OK: 51 MiB in 21 packages
 ---> 81e98c806ee9
Removing intermediate container b057a8183250
Step 3/3 : COPY webserver.py /webserver.py
 ---> c9b7474b12b2
Removing intermediate container 4705922100e6
Successfully built c9b7474b12b2

[jedepuyd@db iox_docker_pythonweb]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ioxpythonweb        1.0                 c9b7474b12b2        11 seconds ago      43.4 MB
alpine              3.3                 461b3f7c318a        2 days ago          4.81 MB
```

La commande de construction de docker télécharge l'image de base et installe le python et les dépendances, en tant que vous ont demandé dans le Dockerfile. La dernière commande est pour la vérification.

5. Testez le conteneur créé de docker.

Cette étape est facultative mais il est bon de vérifier que votre conteneur construit juste de docker est prêt à fonctionner comme prévu.

```
[jedepuyd@db iox_docker_pythonweb]$ docker run -ti ioxpythonweb:1.0
/ # python /webserver.py 9000 &
/ # Starting webserver...

/ # netstat -tlpn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:9000            0.0.0.0:*               LISTEN      7/python
/ # exit
```

Comme vous pouvez voir dans la sortie du netstat, après que vous commenciez le webserver.py, il écoute sur le port 9000.

6. Créez le module d'IOx avec le conteneur de docker.

Maintenant que vous avez vérifié la fonctionnalité de votre web server dans le conteneur, il est temps de préparer et établir le module d'IOx pour le déploiement. Pendant que le Dockerfile fournit des instructions de construire un conteneur de docker, le package.yaml fournit des instructions pour que le client d'IOx établisse votre module d'IOx.

```
jedepuyd@db iox_docker_pythonweb]$ vi package.yaml
[jedepuyd@db iox_docker_pythonweb]$ cat package.yaml
descriptor-schema-version: "2.2"

info:
  name: "iox_docker_pythonweb"
  description: "simple docker python webserver on port 9000"
  version: "1.0"
  author-link: "http://www.cisco.com"
  author-name: "Jens Depuydt"

app:
  cpuarch: "x86_64"
  type: docker
  resources:
    profile: c1.small
    network:
      -
        interface-name: eth0
        ports:
            tcp: [9000]

  startup:
    rootfs: rootfs.tar
    target: ["python","/webserver.py","9000"]
```

Plus d'informations sur le contenu du package.yaml peuvent être trouvées ici : https://developer.cisco.com/media/iox-dev-guide-3-10-16/concepts/package_descriptor/.

Après que vous créiez le package.yaml, vous pouvez commencer à établir le module d'IOx.

La première étape est d'exporter la racine FS de l'image de docker :

```
[jedepuyd@db iox_docker_pythonweb]$ docker save -o rootfs.tar ioxpythonweb:1.0
```

Ensuite, vous pouvez construire package.tar :

```
[jedepuyd@db iox_docker_pythonweb]$ ../ioxclient_1.3.0.0_linux_amd64/ioxclient package .
Currently active profile:  default
Command Name: package
Checking if package descriptor file is present.
Validating descriptor file /home/jedepuyd/iox_docker_pythonweb/package.yaml with package schema
definitions
Parsing descriptor file.
Found schema version  2.2
Loading schema file for version  2.2
Validating package descriptor file..
File /home/jedepuyd/iox_docker_pythonweb/package.yaml is valid under schema version 2.2
Created Staging directory at :  /tmp/700740789
Copying contents to staging directory
Checking for application runtime type
Couldn't detect application runtime type
Creating an inner envelope for application artifacts
Generated  /tmp/700740789/artifacts.tar.gz
Calculating SHA1 checksum for package contents..
Parsing Package Metadata file :  /tmp/700740789/.package.metadata
Wrote package metadata file :  /tmp/700740789/.package.metadata
Root Directory :  /tmp/700740789
```

```
Output file:  /tmp/335805072
Path:  .package.metadata
SHA1 : 55614e72481a64726914b89801a3276a855c728a
Path:  artifacts.tar.gz
SHA1 : 816c7bbfd8ae76af451642e652bad5cf9592370c
Path:  package.yaml
SHA1 : ae75859909f6ea6947f599fd77a3f8f04fda0709
Generated package manifest at  package.mf
Generating IOx Package..
Package generated at /home/jedepuyd/iox_docker_pythonweb/package.tar
```

Le résultat de la construction est un module d'IOx (package.tar), qui contient le conteneur de docker, prêt à être déployé sur IOx.

> Remarque: IOxclient peut faire la commande de sauvegarde de docker dans une étape aussi bien. Sur CentOS, ceci résulte pour exporter au par défaut rootfs.img au lieu de rootfs.tar, qui donne le problème plus tard dans le processus. L'une étape à créer peut être faite avec l'utilisation de : Module IOxpythonweb:1.0 de docker de client d'IOx.

8. Déployez, lancez et commencez le module sur le périphérique d'IOx.

Les dernières étapes sont de déployer le module d'IOx vers le périphérique d'IOx, lancer lui et le début. Ces étapes peuvent être faites avec l'utilisation du client d'IOx, du gérant local ou du directeur de réseau de brouillard. Pour cet article, vous pouvez utiliser le client d'IOx.

Afin de déployer le module vers le périphérique d'IOx, python_web de nom d'utilisation :

```
[jedepuyd@db iox_docker_pythonweb]$ ../ioxclient_1.3.0.0_linux_amd64/ioxclient app install
python_web package.tar
Currently active profile:  default
Command Name: application-install
Installation Successful. App is available at:
https://10.48.43.197:8443/iox/api/v2/hosting/apps/python_web
Successfully deployed
```

Avant que vous puissiez lancer l'application, vous devez définir comment la configuration réseau serait. Pour faire ainsi, vous devez créer un fichier JSON. Quand vous lancez, il peut être relié à la demande de lancement.

```
[jedepuyd@db iox_docker_pythonweb]$ vi activate.json
[jedepuyd@db iox_docker_pythonweb]$ cat activate.json
{
    "resources": {
        "profile": "c1.small",
        "network": [{"interface-name": "eth0", "network-name": "iox-nat0"}]
    }
}
[jedepuyd@db iox_docker_pythonweb]$ ../ioxclient_1.3.0.0_linux_amd64/ioxclient app activate
python_web --payload activate.json
Currently active profile :  default
Command Name: application-activate
Payload file : activate.json. Will pass it as application/json in request body..
App python_web is Activated
```

La dernière action ici est de commencer l'application que vous avez juste déployée et avez lancée :

```
[jedepuyd@db iox_docker_pythonweb]$ ../ioxclient_1.3.0.0_linux_amd64/ioxclient app start
python_web
Currently active profile :  default
```

```
Command Name: application-start
App python_web is Started
```

Puisque vous avez configuré votre application d'IOx pour écouter sur le port 9000 des HTTP-demandes de massif de roche, vous devez toujours expédier ce port de votre IOx-périphérique dans le conteneur pendant que le conteneur est derrière NAT. Exécutez ceci sur le Cisco IOS® pour faire ainsi.

```
BRU-IOT-809-1#sh iox host list det | i IPV4
    IPV4 Address of Host:       192.168.1.2
BRU-IOT-809-1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
BRU-IOT-809-1(config)#ip nat inside source static tcp 192.168.1.2 9000 interface
GigabitEthernet0 9000
BRU-IOT-809-1(config)#exit
```

Les premières listes de commandes l'adresse IP interne du gos (responsable de début et de fin/exécutez les conteneurs d'IOx).

La deuxième commande configure une prise de pression statique en avant pour le port 9000 sur l'interface Gi0 du l'IOS-side au gos. Au cas où votre périphérique serait connecté par l'intermédiaire d'un port L2 (qui est le plus susceptible le cas sur IR829), vous devez remplacer l'interface Gi0 par le VLAN correct qui a la déclaration d'ip nat outside configurée.

Vérifiez

Utilisez cette section pour confirmer que votre configuration fonctionne correctement.

Afin de vérifier si le web server fonctionne et répond correctement, vous pouvez essayer d'accéder au web server avec cette commande.

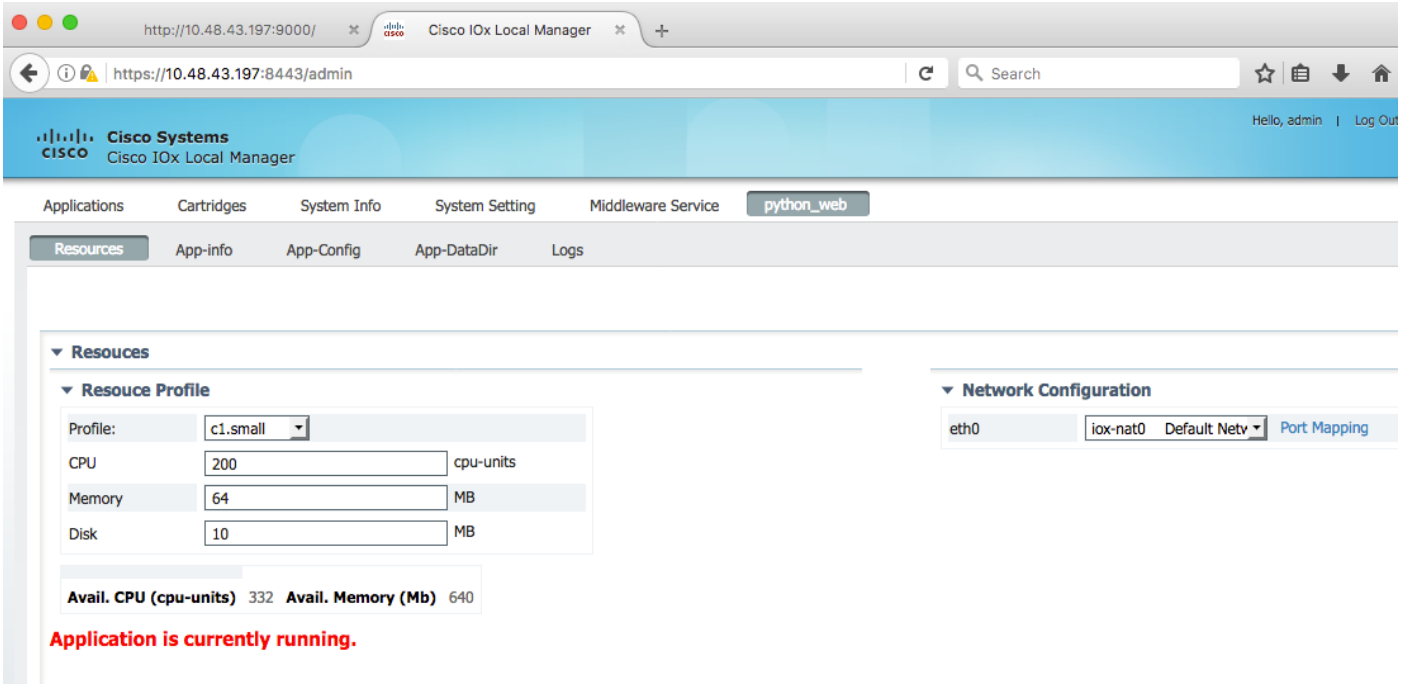
```
[jedepuyd@db iox_docker_pythonweb]$ curl http://10.48.43.197:9000/
<html><body><h1>IOX python webserver</h1></body></html>
```

Ou, d'un vrai navigateur suivant les indications de l'image.

Vous pouvez également vérifier l'état d'application de l'IOxclient CLI :

```
[jedepuyd@db iox_docker_pythonweb]$ ../ioxclient_1.3.0.0_linux_amd64/ioxclient app status
python_web
Currently active profile :  default
Command Name: application-status
Saving current configuration
App python_web is RUNNING
```

et vous pouvez également vérifier l'état d'application du GUI de gestionnaire local suivant les indications de l'image.



Afin d'aller voir un regarder le fichier journal que vous écrivez à dans webserver.py :

```
[jedepuyd@db iox_docker_pythonweb]$ ../ioxclient_1.3.0.0_linux_amd64/ioxclient app logs info
python_web
Currently active profile :  default
Command Name: application-logs-info

Log file information for :  python_web
Size_bytes : 711
Download_link : /admin/download/logs?filename=python_web-watchDog.log
Timestamp : Thu Jun 22 08:21:18 2017
Filename : watchDog.log

Size_bytes : 23
Download_link : /admin/download/logs?filename=python_web-webserver.log
Timestamp : Thu Jun 22 08:21:23 2017
Filename : webserver.log

Size_bytes : 2220
Download_link : /admin/download/logs?filename=python_web-container_log_python_web.log
Timestamp : Thu Jun 22 08:21:09 2017
Filename : container_log_python_web.log
```

Dépannez

Cette section fournit des informations que vous pouvez utiliser pour dépanner votre configuration.

Afin de dépanner l'application et/ou le conteneur, le moyen le plus simple est de se connecter à la console de l'application qui fonctionne :

```
[jedepuyd@db iox_docker_pythonweb]$ ../ioxclient_1.3.0.0_linux_amd64/ioxclient app console
python_web
Currently active profile:  default
Command Name: application-console
Console setup is complete..
Running command: [ssh -p 2222 -i python_web.pem appconsole@10.48.43.197]
The authenticity of host '[10.48.43.197]:2222 ([10.48.43.197]:2222)' can't be established.
ECDSA key fingerprint is 1d:e4:1e:e1:99:8b:1d:d5:ca:43:69:6a:a3:20:6d:56.
Are you sure you want to continue connecting (yes/no)? yes
```

```
/ # netstat -tlpn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:9000            0.0.0.0:*               LISTEN      19/python
/ # ps aux | grep python
   19 root       0:00 python /webserver.py 9000
```