Configurar una pequeña imagen de acoplador de Linux alpino en IOx

Contenido

Introducción

Este documento describe el proceso de configuración para crear, implementar y administrar aplicaciones basadas en Docker en dispositivos compatibles con Cisco IOx.

Prerequisites

Requirements

No hay requisitos específicos para este documento.

Componentes Utilizados

La información que contiene este documento se basa en las siguientes versiones de software y hardware.

- Dispositivo compatible con E/Sx configurado para E/Sx:
  Dirección IP configuradaSistema operativo invitado (GOS) y Cisco Application Framework (CAF) en ejecuciónTraducción de direcciones de red (NAT) configurada para el acceso a CAF (puerto 8443)NAT configurado para acceso al shell GOS (puerto 2222)
- Host Linux (se utiliza una instalación mínima de CentOS 7 para este artículo)
- Archivos de instalación del cliente IOx que se pueden descargar desde: https://software.cisco.com/download/release.html?mdfid=286306005&softwareid=286306762

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Antecedentes

IOx puede alojar diferentes tipos de paquetes principalmente Java, Python, LXC, máquina virtual

(VM), etc, y también puede ejecutar contenedores Docker. Cisco ofrece una imagen básica y un repositorio completo del concentrador Docker: https://devhub.cisco.com/artifactory/webapp/#/artifacts/browse/tree/General/iox-docker que se puede utilizar para construir contenedores Docker.

Esta es una guía paso a paso sobre cómo construir un simple contenedor de Docker con el uso de Alpine Linux. Alpine Linux es una pequeña imagen Linux (alrededor de 5 MB), que a menudo se utiliza como base para los contenedores Docker. En este artículo, se inicia desde un dispositivo IOx configurado, una máquina Linux CentOS 7 vacía y se crea un servidor web Python pequeño, se empaqueta en un contenedor Docker e implementa eso en un dispositivo IOx.

Configurar

1. Instale y prepare el cliente IOx en el host Linux.

El cliente IOx es la herramienta que puede empaquetar aplicaciones y comunicarse con el dispositivo compatible con IOx para administrar aplicaciones IOx.

Después de descargar el paquete de instalación de ioxclient, se puede instalar de la siguiente manera:

```
[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
```

Como puede ver, en el primer lanzamiento del cliente IOx, se puede generar un perfil para el dispositivo IOx que puede administrar con el cliente IOx. En caso de que desee hacerlo más tarde o si desea agregar o cambiar la configuración, puede ejecutar este comando más tarde: **ioxclient profiles create**

2. Instale y prepare Docker en el host Linux.

Docker se utiliza para generar un contenedor y probar la ejecución de nuestra aplicación de

ejemplo.

Los pasos de instalación para instalar Docker dependen en gran medida del sistema operativo Linux en el que se instala. Para este artículo, puede utilizar CentOS 7. Para obtener instrucciones de instalación para diferentes distribuciones, consulte: https://docs.docker.com/engine/installation/.

Instalar requisitos previos:

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

Agregue la 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
```

Instale el acoplador (acepte la verificación de la clave GPG cuando instale):

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

Iniciar acoplador:

```
[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
```

Para poder acceder/ejecutar Docker como usuario normal, agregue este usuario al grupo Docker y actualice la pertenencia al grupo:

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

Inicie sesión en Docker Hub:

El Docker Hub contiene la imagen base alpina que puede utilizar. En caso de que aún no tenga una ID de Docker, deberá registrarse en: 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. Cree el servidor web de Python.

Ahora que se ha realizado la preparación, puede comenzar a generar la aplicación real que se puede ejecutar en el dispositivo de habilitación IOx.

```
[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()
```

Este código es un servidor web Python muy mínimo, que se crea en webserver.py. El servidor web simplemente devuelve el servidor web IOx python tan pronto como se solicita un GET. El puerto en el que se inicia el servidor web puede ser el puerto 80 o el primer argumento dado a webserver.py.

Este código también contiene, en la función de ejecución, una escritura en un archivo de registro. El archivo de registro está disponible para consulta del cliente IOx o del administrador local.

4. Cree el archivo Dockerfile y el contenedor Docker.

Ahora que tiene la aplicación (webserver.py) que debe ejecutarse en su contenedor, es hora de crear el contenedor Docker. Un contenedor se define en un archivo 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
```

Como puede ver, el archivo Dockerfile también se mantiene sencillo. Comience con la imagen base alpina, instale Python y copie su webserver.py a la raíz del contenedor.

Una vez que haya preparado el archivo Dockerfile, puede generar el contenedor 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
```

El comando Docker build descarga la imagen base e instala Python y dependencias, como solicitó en el archivo Dockerfile. El último comando es para verificación.

5. Pruebe el contenedor de Docker creado.

Este paso es opcional pero es bueno verificar que el contenedor de Docker recién construido esté listo para funcionar como se espera.

```
[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
```

Como puede ver en el resultado de netstat, después de iniciar webserver.py, escucha en el puerto 9000.

6. Cree el paquete IOx con el contenedor Docker.

Ahora que ha verificado la funcionalidad de su servidor web en el contenedor, ha llegado el momento de preparar y crear el paquete IOx para la implementación. Como el archivo Dockerfile proporciona instrucciones para construir un contenedor Docker, package.yaml proporciona instrucciones para que el cliente IOx construya su paquete 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"]
```

Puede encontrar más información sobre el contenido de package.yaml aquí: https://developer.cisco.com/media/iox-dev-guide-3-10-16/concepts/package_descriptor/.

Después de crear el package.yaml, puede comenzar a generar el paquete IOx.

El primer paso es exportar el FS raíz de la imagen Docker:

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

A continuación, puede generar el archivo 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
```

El resultado de la generación es un paquete IOx (package.tar), que contiene el contenedor Docker, listo para ser implementado en IOx.

> **Nota:** IOxclient también puede realizar el comando docker save en un paso. En CentOS, esto resulta en exportar a los rootfs.img predeterminados en lugar de rootfs.tar, lo que da problemas más adelante en el proceso. El único paso a crear se puede realizar con el uso de: IOxpythonweb:1.0 paquete de docker de cliente IOx.

8. Implemente, active e inicie el paquete en el dispositivo IOx.

Los últimos pasos son implementar el paquete IOx en el dispositivo IOx, activarlo e iniciarlo. Estos pasos se pueden realizar con el uso del cliente IOx, Local Manager o Fog Network Director. Para este artículo, puede utilizar el cliente IOx.

Para implementar el paquete en el dispositivo IOx, utilice name python_web:

```
[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
```

Antes de activar la aplicación, debe definir la configuración de la red. Para ello, debe crear un archivo JSON. Cuando se activa, se puede adjuntar a la solicitud de activación.

```
[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","port_map": {"mode":
"1to1"},"ports":{"tcp":9000}}]
    }
}
[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 última acción aquí es iniciar la aplicación que acaba de implementar y activar:

```
[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
```

Dado que configuró su aplicación IOx para escuchar en el puerto 9000 las solicitudes tor HTTP, todavía necesita reenviar ese puerto desde su dispositivo IOx al contenedor como el contenedor está detrás de NAT. Realice esto en Cisco IOS® para hacerlo.

```
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
```

El primer comando enumera la dirección IP interna de GOS (responsable de iniciar/detener/ejecutar los contenedores IOx).

El segundo comando configura un puerto estático hacia adelante para el puerto 9000 en la interfaz Gi0 del lado del IOS hacia GOS. En caso de que su dispositivo esté conectado a través de un puerto L2 (lo que probablemente sea el caso en IR829), debe reemplazar la interfaz Gi0 por la VLAN correcta que tiene configurada la sentencia externa ip nat.

Verificación

Utilize esta sección para confirmar que su configuración funcione correctamente.

Para verificar si el servidor web se ejecuta y responde correctamente, puede intentar acceder al servidor web con este comando.

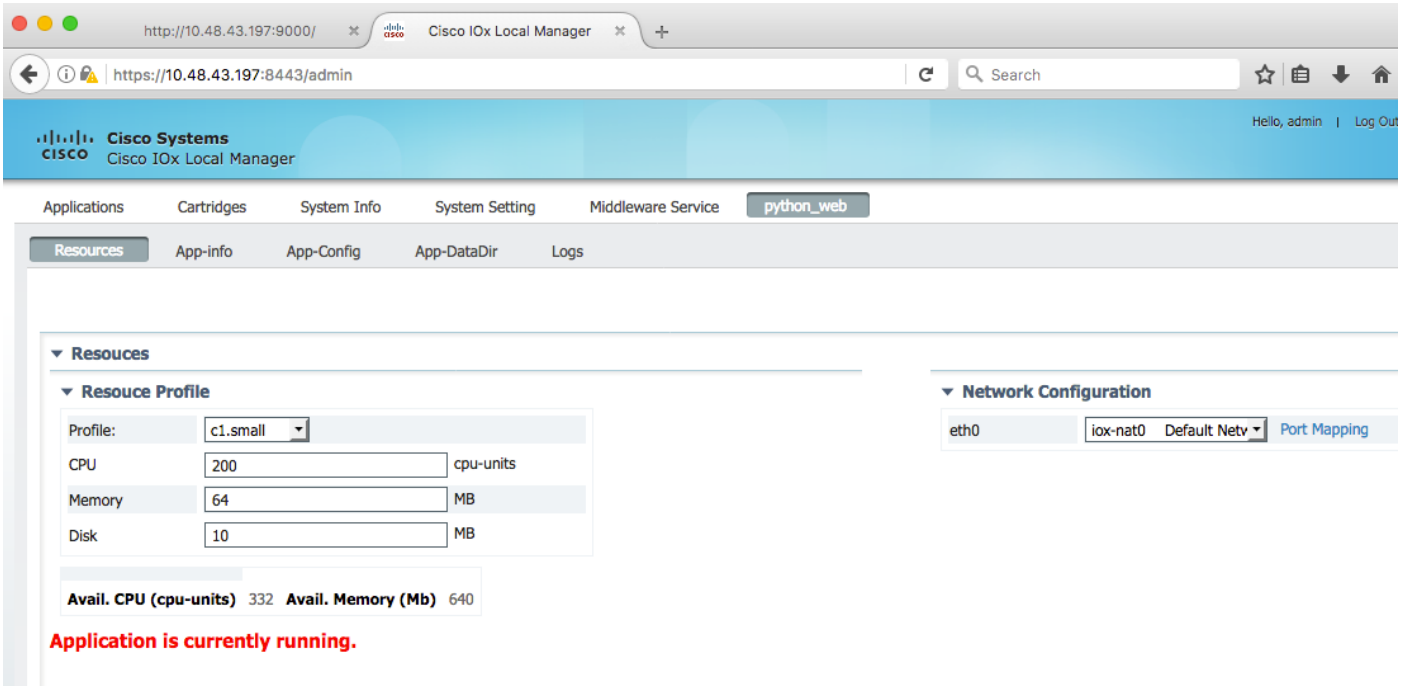
```
[jedepuyd@db iox_docker_pythonweb]$ curl http://10.48.43.197:9000/
<html><body><h1>IOX python webserver</h1></body></html>
```

O bien, desde un explorador real como se muestra en la imagen.

También puede verificar el estado de la aplicación desde la CLI de Exclient:

```
[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
```

y también puede verificar el estado de la aplicación desde la GUI del administrador local, como se muestra en la imagen.



Para ver el archivo de registro al que escribe en 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
```

Troubleshoot

En esta sección se brinda información que puede utilizar para resolver problemas en su configuración.

Para resolver problemas de la aplicación y/o contenedor, la manera más fácil es conectarse a la consola de la aplicación que se ejecuta:

```
[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
```