# Configure una pequeña imagen alpina del estibador de Linux en IOx

## Contenido



## Introducción

Este documento describe el proceso de configuración para crear, para desplegar y para manejar las aplicaciones Estibador-basadas en los dispositivos IOx-capaces de Cisco.

## Prerrequisitos

### Requisitos

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 con capacidad de IOx que se configura para IOx:
  Dirección IP configuradaSistema operativo del invitado (GOS) y funcionamiento del marco de la aplicación de Cisco (CAF)Network Address Translation (NAT) configurado para el acceso a CAF (puerto 8443)NAT configurado para el acceso al shell GOS (puerto 2222)
- Host de Linux (una instalación mínima de CentOS 7 se utiliza para este artículo)
- Archivos de la instalación del cliente de IOx de los cuales puede ser descargado: https://software.cisco.com/download/release.html?mdfid=286306005&softwareid=286306762

La información que contiene este documento se creó a partir de los dispositivos en un ambiente de laboratorio específico. Todos los dispositivos que se utilizan en este documento se pusieron en funcionamiento con una configuración verificada (predeterminada). Si la red está funcionando, asegúrese de haber comprendido el impacto que puede tener cualquier comando.

## Antecedentes

IOx puede recibir diversos tipos de las Javas de los paquetes principalmente, de Python, de LXC, de la máquina virtual (VM) etc, y puede también funcionar con los envases del estibador. Cisco ofrece una imagen base y un repositorio completo del concentrador del estibador: https://devhub.cisco.com/artifactory/webapp/#/artifacts/browse/tree/General/iox-docker que se puede utilizar para construir los envases del estibador.

Aquí está un guía paso a paso en cómo construir un envase simple del estibador con el uso de Linux alpino. Linux alpino es una pequeña imagen de Linux (alrededor de 5MB), que es de uso frecuente como base para los envases del estibador. En este artículo, usted sale de un dispositivo configurado de IOx, una máquina y usted vacíos de CentOS 7 Linux construyen a un pequeño servidor Web de Python, lo empaquetan en un envase del estibador y despliegan eso en un dispositivo de IOx.

# Configurar

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

El cliente de IOx es la herramienta que puede empaquetar las aplicaciones y comunicar con el dispositivo IOx-capaz para manejar las aplicaciones de IOx.

Después de que usted descargue el paquete ioxclient de la instalación, puede ser instalado como sigue:

```
[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 usted puede ver, en el primer lanzamiento del cliente de IOx, un perfil puede ser generado para el dispositivo de IOx que usted puede manejar con el cliente de IOx. En caso de que usted quisiera hacer esto más adelante o si usted quiere agregar/cambio las configuraciones, usted puede funcionar con este comando más adelante: **los perfiles ioxclient crean**

2. Instale y prepare al estibador en el host de Linux.

Utilizan al estibador para construir un envase y para probar la ejecución de nuestra aplicación de ejemplo.

Los pasos de la instalación para instalar al estibador dependen pesadamente del Linux OS que usted lo instala encendido. Para este artículo, usted puede utilizar CentOS 7. Para las instrucciones de instalación para diversas distribuciones,
refiérase: https://docs.docker.com/engine/installation/.

Instale los requisitos previos:

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

Agregue el repo del estibador:

```
[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 al estibador (valide la verificación de la clave GPG cuando usted instala):

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

Estibador del comienzo:

```
[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/estibador funcionado con como usuario común, agregue a este usuario al grupo del estibador y restaure la membresía del grupo:

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

Inicie sesión al concentrador del estibador:

El concentrador del estibador contiene la imagen base alpina que usted puede utilizar. En caso de que usted no tenga un estibador ID todavía, usted necesita registrarse encendido: 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 al servidor Web de Python.

Ahora que se hace la preparación, usted puede comenzar a construir la aplicación real que puede ejecutarse en el dispositivo del IOx-permiso.

```
[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 muy mínimo de Python, que usted crea en webserver.py. El servidor Web vuelve simplemente el web server del pitón de IOx tan pronto como se pida un GET. El puerto en el cual el servidor Web comienza puede ser el puerto 80 o el primer argumento dado a webserver.py.

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

4. Cree el envase de Dockerfile y del estibador.

Ahora que usted tiene la aplicación (webserver.py) que debe ejecutarse en su envase, es hora de construir el envase del estibador. Un envase se define en 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
```

Como usted puede ver, el Dockerfile también se mantiene simple. Usted comienza con la imagen base alpina, instala Python y copia su webserver.py a la raíz del envase.

Una vez que usted tiene su Dockerfile listo, usted puede construir el envase del estibador:

```
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 de la estructura del estibador descarga la imagen base y instala Python y las dependencias, como usted pidieron en el Dockerfile. El comando más reciente está para la verificación.

5. Pruebe el envase creado del estibador.

Este paso es opcional pero es bueno verificar que su envase construido justo del estibador está listo para trabajar como se esperaba.

```
[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 usted puede ver en la salida del netstat, después de que usted comience el webserver.py, escucha en el puerto 9000.

6. Cree el paquete de IOx con el envase del estibador.

Ahora que usted ha verificado las funciones de su servidor Web en el envase, es hora de preparar y de construir el paquete de IOx para el despliegue. Mientras que el Dockerfile proporciona las instrucciones de construir un envase del estibador, el package.yaml proporciona las instrucciones para que el cliente de IOx construya su paquete de 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"]
```

Más información sobre el contenido del package.yaml se puede encontrar aquí: https://developer.cisco.com/media/iox-dev-guide-3-10-16/concepts/package_descriptor/.

Después de que usted cree el package.yaml, usted puede comenzar a construir el paquete de IOx.

El primer paso es exportar la raíz FS de la imagen del estibador:

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

Después, usted puede construir 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 estructura es un paquete de IOx (package.tar), que contiene el envase del estibador, listo para ser desplegado en IOx.

> Nota: IOxclient puede hacer el comando save del estibador en un paso también. En CentOS, esto resulta para exportar al valor por defecto rootfs.img en vez de rootfs.tar, que da el problema más adelante en el proceso. El un paso a crear se puede hacer con el uso de: Paquete IOxpythonweb:1.0 del estibador del cliente de IOx.

8. Despliegue, active y comience el paquete en el dispositivo de IOx.

Los pasos más recientes son desplegar el paquete de IOx al dispositivo de IOx, activarlo y comenzar. Estos pasos se pueden hacer con el uso del cliente de IOx, del administrador local o del director de la red de la niebla. Para este artículo, usted puede utilizar al cliente de IOx.

Para desplegar el paquete al dispositivo de IOx, utilice el python_web del nombre:

```
[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 que usted pueda activar la aplicación, usted necesita definir cómo sería la configuración de red. Para hacer así pues, usted necesita crear un archivo JSON. Cuando usted activa, puede ser asociado a la petición de la 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"}]
    }
}
[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 acción más reciente aquí es comenzar la aplicación que usted desplegó y acaba de activa:

```
[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
```

Puesto que usted configuró su aplicación de IOx para escuchar en el puerto 9000 los pedidos de HTTP del tor, usted todavía necesita remitir ese puerto de su IOx-dispositivo al envase mientras que el envase está detrás de NAT. Realice esto en Cisco IOS® para hacer tan.

```
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
```

Las primeras listas de comandos el IP Address interno del GOS (responsable de partida/parada/funcione con los envases de IOx).

El comando second configura un puerto estático adelante para el puerto 9000 en la interfaz Gi0 del IOS-lado al GOS. En caso de que su dispositivo esté conectado vía un puerto L2 (que sea más probable el caso en IR829), usted necesita substituir la interfaz Gi0 por el VLA N correcto que tiene la declaración exterior nacional del IP configurada.

## Verificación

Utilize esta sección para confirmar que su configuración funcione correctamente.

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

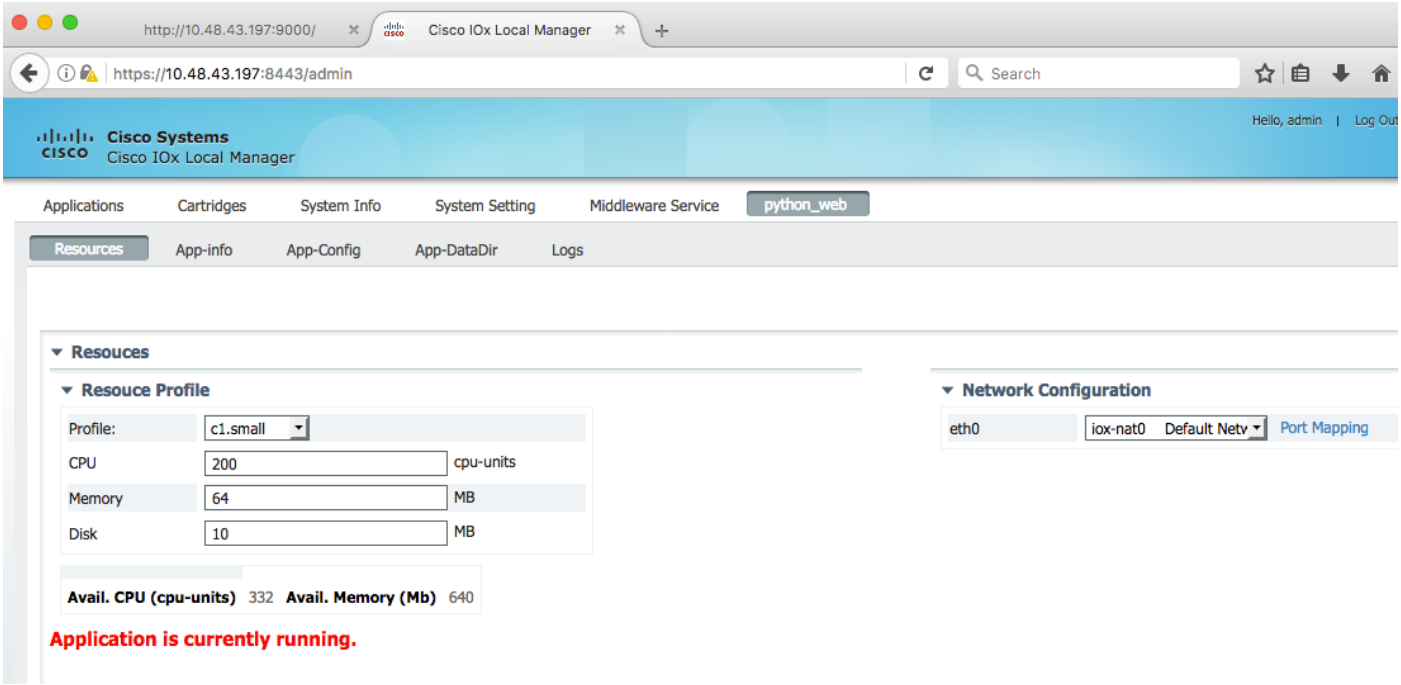
```
[jedepuyd@db iox_docker_pythonweb]$ curl http://10.48.43.197:9000/
<html><body><h1>IOX python webserver</h1></body></html>
```

O, de un navegador real tal y como se muestra en de la imagen.


Usted puede también verificar el estatus de la aplicación del 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
```

y usted puede también verificar el estatus de la aplicación del administrador local GUI tal y como se muestra en de la imagen.




Para tener una mirada en el archivo del registro que usted escribe en a 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
```

# Troubleshooting

Esta sección proporciona la información que usted puede utilizar para resolver problemas su configuración.

Para resolver problemas la aplicación y/o el envase, la manera más fácil es conectar con 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
```