# Configurazione di un'immagine piccola di Alpine Linux Docker su IOx

## Sommario



## Introduzione

Questo documento descrive il processo di configurazione per creare, distribuire e gestire applicazioni basate su Docker su dispositivi compatibili con Cisco IOx.

## Prerequisiti

### Requisiti

Nessun requisito specifico previsto per questo documento.

### Componenti usati

Le informazioni fornite in questo documento si basano sulle seguenti versioni software e hardware:

- Dispositivo compatibile con IOx configurato per IOx:
  Indirizzo IP configuratoSistema operativo guest (GOS) e Cisco Application Framework (CAF) in esecuzioneNetwork Address Translation (NAT) configurato per l'accesso a CAF (porta 8443)NAT configurato per l'accesso alla shell GOS (porta 2222)
- Host Linux (per questo articolo viene utilizzata un'installazione minima di CentOS 7)
- File di installazione del client IOx scaricabili da:
  https://software.cisco.com/download/release.html?mdfid=286306005&softwareid=286306762

Le informazioni discusse in questo documento fanno riferimento a dispositivi usati in uno specifico ambiente di emulazione. Su tutti i dispositivi menzionati nel documento la configurazione è stata ripristinata ai valori predefiniti. Se la rete è operativa, valutare attentamente eventuali conseguenze derivanti dall'uso dei comandi.

## Premesse

IOx può ospitare diversi tipi di pacchetti, principalmente Java, Python, LXC, Virtual Machine (VM) e così via, e può anche eseguire contenitori Docker. Cisco offre un'immagine di base e un repository hub Docker completo: https://devhub.cisco.com/artifactory/webapp/#/artifacts/browse/tree/General/iox-docker che può essere utilizzato per creare contenitori Docker.

Ecco una guida dettagliata su come costruire un semplice contenitore Docker con l'uso di Alpine Linux. Alpine Linux è un'immagine Linux di piccole dimensioni (circa 5 MB), spesso utilizzata come base per i contenitori Docker. In questo articolo, si parte da un dispositivo IOx configurato, un computer Linux CentOS 7 vuoto e si crea un piccolo server Web Python, lo si racchiude in un contenitore Docker e lo si distribuisce su un dispositivo IOx.

# Configurazione

1. Installare e preparare il client IOx sull'host Linux.

Il client IOx è lo strumento in grado di creare pacchetti di applicazioni e comunicare con il dispositivo compatibile con IOx per gestire le applicazioni IOx.

Dopo aver scaricato il pacchetto di installazione di ioxclient, è possibile installarlo come segue:

```
[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
```

Come si può vedere, al primo avvio del client IOx è possibile generare un profilo per il dispositivo IOx che è possibile gestire con il client IOx. Se si desidera eseguire questa operazione in un secondo momento o aggiungere/modificare le impostazioni, è possibile eseguire questo comando in un secondo momento: **ioxclient, profili creare**

2. Installare e preparare Docker sull'host Linux.

Docker viene utilizzato per creare un contenitore e testare l'esecuzione dell'applicazione di

esempio.

La procedura di installazione di Docker dipende in larga misura dal sistema operativo Linux su cui viene installato. Per questo articolo, è possibile utilizzare CentOS 7. Per istruzioni di installazione per distribuzioni diverse, fare riferimento a: https://docs.docker.com/engine/installation/.

Installa prerequisiti:

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

Aggiungere il repository 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
```

Installa Docker (accetta la verifica del codice GPG al momento dell'installazione):

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

Avvia Docker:

```
[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
```

Per poter accedere/eseguire Docker come utente normale, aggiungere questo utente al gruppo Docker e aggiornare l'appartenenza al gruppo:

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

Accedere a Docker Hub:

L'hub Docker contiene l'immagine base Alpina che è possibile utilizzare. Se non si dispone ancora di un ID Docker, è necessario registrarsi su: 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. Creare il server Web Python.

Al termine della preparazione, è possibile iniziare a creare l'applicazione effettiva che può essere eseguita sul dispositivo abilitato per 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()
```

Questo codice è un server Web Python molto piccolo, creato in webserver.py. Il server Web restituisce semplicemente il server Web IOx Python non appena viene richiesto un GET. La porta di avvio del server Web può essere la porta 80 o il primo argomento assegnato a webserver.py.

Questo codice contiene inoltre, nella funzione run, una scrittura in un file di log. Il file di registro è disponibile per la consultazione dal client IOx o dal responsabile locale.

4. Creare il file Docker e il contenitore Docker.

Ora che si dispone dell'applicazione (webserver.py) che dovrebbe essere eseguita nel contenitore, è necessario compilare il contenitore Docker. Un contenitore è definito in 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
```

Come si può vedere, anche il Dockerfile è mantenuto semplice. Si inizia con l'immagine base Alpine, si installa Python e si copia il webserver.py alla radice del contenitore.

Una volta pronto il file Docker, è possibile creare il contenitore 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
```

Il comando Docker build scarica l'immagine di base e installa Python e le dipendenze, come richiesto nel file Docker. L'ultimo comando serve per la verifica.

5. Verificare il contenitore Docker creato.

Questo passaggio è facoltativo, ma è consigliabile verificare che il contenitore Docker appena creato sia pronto per funzionare come previsto.

```
[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
```

Come si può vedere nell'output di netstat, dopo aver avviato webserver.py, questo resta in ascolto sulla porta 9000.

6. Creare il pacchetto IOx con il contenitore Docker.

Dopo aver verificato la funzionalità del server Web nel contenitore, è possibile preparare e generare il pacchetto IOx per la distribuzione. Poiché il file Docker fornisce istruzioni per la creazione di un contenitore Docker, package.yaml fornisce istruzioni per il client IOx per la creazione del pacchetto 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"]
```

Per ulteriori informazioni sul contenuto del package.yaml, consultare il sito: https://developer.cisco.com/media/iox-dev-guide-3-10-16/concepts/package_descriptor/.

Dopo aver creato il package.yaml, è possibile iniziare a generare il package IOx.

Il primo passaggio consiste nell'esportare il file system radice dell'immagine Docker:

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

Successivamente, è possibile generare il file 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
```

Il risultato della compilazione è un pacchetto IOx (package.tar), che contiene il contenitore Docker, pronto per essere distribuito in IOx.

> **Nota:** IOxclient può eseguire il comando docker save in un unico passaggio. In CentOS, il risultato è l'esportazione nel file rootfs.img predefinito invece che nel file rootfs.tar, che causa problemi più avanti nel processo. L'unico passaggio da creare può essere fatto con l'uso di: IOx client docker package IOxpythonweb:1.0.

8. Distribuire, attivare e avviare il pacchetto sul dispositivo IOx.

Gli ultimi passaggi consistono nel distribuire il pacchetto IOx nel dispositivo IOx, attivarlo e avviarlo. Queste operazioni possono essere eseguite con il client IOx, con Local Manager o con Fog Network Director. Per questo articolo, è possibile utilizzare il client IOx.

Per distribuire il pacchetto al dispositivo IOx, utilizzare il nome 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
```

Prima di attivare l'applicazione, è necessario definire la configurazione di rete. A tale scopo, è necessario creare un file JSON. Quando si esegue l'attivazione, è possibile allegarla alla richiesta di attivazione.

```
[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
```

L'ultima azione consiste nell'avviare l'applicazione appena distribuita e attivata:

```
[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
```

Poiché l'applicazione IOx è stata configurata per l'ascolto sulla porta 9000 per le richieste HTTP o, è comunque necessario inoltrare la porta dal dispositivo IOx al contenitore, poiché il contenitore si trova dietro NAT. Eseguire questa operazione su Cisco IOS®.

```
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
```

Il primo comando elenca l'indirizzo IP interno di GOS (responsabile dell'avvio, dell'arresto e dell'esecuzione dei contenitori IOx).

Il secondo comando configura una porta statica in avanti per la porta 9000 sull'interfaccia Gi0 del lato IOS su GOS. Se il dispositivo è collegato tramite una porta L2 (come nel caso di IR829), è necessario sostituire l'interfaccia Gi0 con la VLAN corretta in cui è configurata l'istruzione ip nat outside.

# Verifica

Fare riferimento a questa sezione per verificare che la configurazione funzioni correttamente.

Per verificare se il server Web viene eseguito e risponde correttamente, è possibile provare ad accedere al server Web con questo comando.

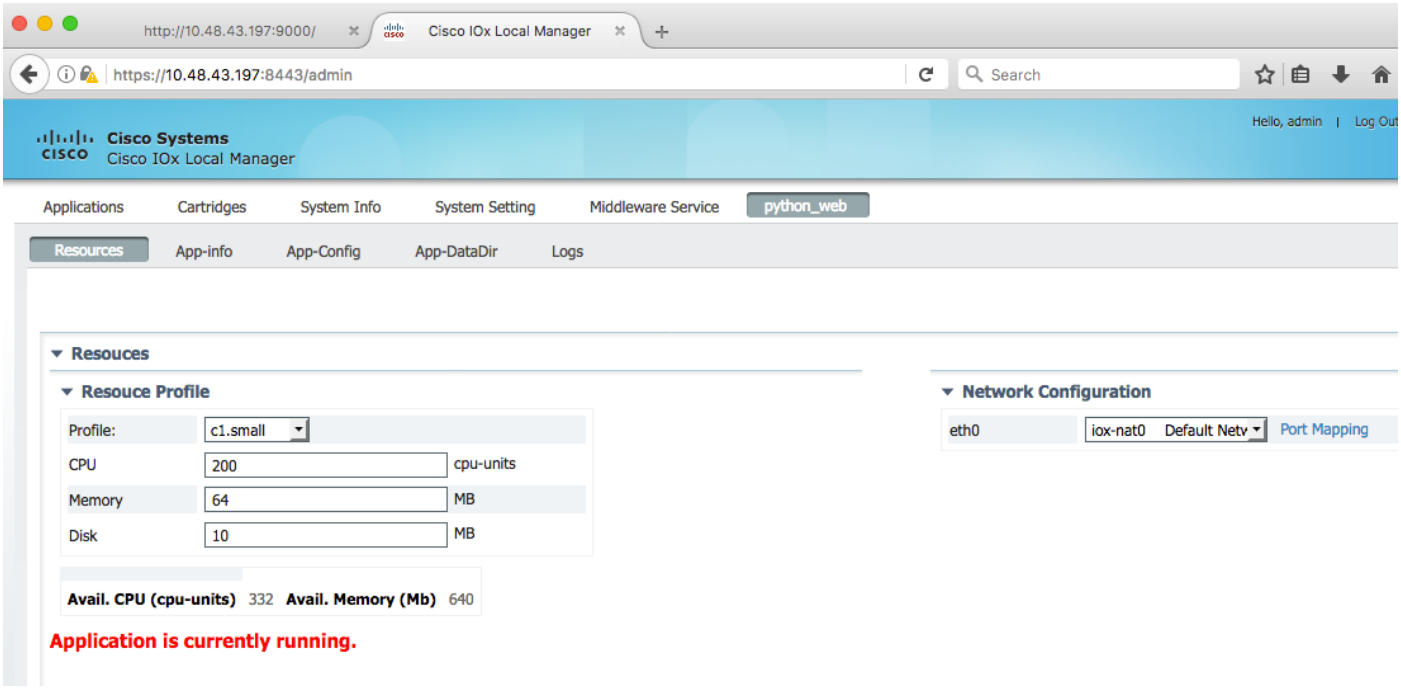
```
[jedepuyd@db iox_docker_pythonweb]$ curl http://10.48.43.197:9000/
<html><body><h1>IOX python webserver</h1></body></html>
```

Oppure da un browser reale come mostrato nell'immagine.


Èinoltre possibile verificare lo stato dell'applicazione dalla CLI di IOxclient:

```
[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
```

èinoltre possibile verificare lo stato dell'applicazione dall'interfaccia utente di Local Manager, come mostrato nell'immagine.




Per esaminare il file di registro in cui si scrive in 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
```

# Risoluzione dei problemi

Le informazioni contenute in questa sezione permettono di risolvere i problemi relativi alla configurazione.

Per risolvere i problemi relativi all'applicazione e/o al contenitore, il modo più semplice consiste nel connettersi alla console dell'applicazione in esecuzione:

```
[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
```