

IOx에서 소규모 Alpine Linux Docker 이미지 구성

목차

[소개](#)

[사전 요구 사항](#)

[요구 사항](#)

[사용되는 구성 요소](#)

[배경 정보](#)

[구성](#)

[다음을 확인합니다.](#)

[문제 해결](#)

소개

이 문서에서는 Cisco IOx 지원 디바이스에서 Docker 기반 애플리케이션을 생성, 구축 및 관리하기 위한 컨피그레이션 프로세스에 대해 설명합니다.

사전 요구 사항

요구 사항

이 문서에 대한 특정 요건이 없습니다.

사용되는 구성 요소

이 문서의 정보는 다음 소프트웨어 및 하드웨어 버전을 기반으로 합니다.

- IOx용으로 구성된 IOx 지원 디바이스:
 - 구성된 IP 주소
 - 게스트 GOS(Operating System) 및 Cisco CAF(Application Framework) 실행
 - CAF(포트 8443)에 액세스하도록 구성된 NAT(Network Address Translation)
 - GOS 셸에 액세스하도록 구성된 NAT(포트 2222)
- Linux 호스트(이 문서에는 최소 CentOS 7 설치가 사용됨)
- 다운로드 가능한 IOx 클라이언트 설치 파일:

<https://software.cisco.com/download/release.html?mdfid=286306005&softwareid=286306762>

이 문서의 정보는 특정 랩 환경의 디바이스를 토대로 작성되었습니다. 이 문서에 사용된 모든 디바이스는 초기화된(기본) 컨피그레이션으로 시작되었습니다. 현재 네트워크가 작동 중인 경우, 모든 명령어의 잠재적인 영향을 미리 숙지하시기 바랍니다.

배경 정보

IOx는 Java, Python, LXC, VM(Virtual Machine) 등 다양한 유형의 패키지를 호스팅할 수 있으며 Docker 컨테이너도 실행할 수 있습니다. Cisco는 기본 이미지와 Docker 컨테이너를 구축하는 데 사용할 수 있는 [완전한 Docker 허브 저장소](https://devhub.cisco.com/artifactory/webapp/#/artifacts/browse/tree/General/iox-docker) (<https://devhub.cisco.com/artifactory/webapp/#/artifacts/browse/tree/General/iox-docker>)를 제공합니다.

다음은 Alpine Linux를 사용하여 간단한 Docker 컨테이너를 구축하는 방법에 대한 단계별 가이드입니다. Alpine Linux는 소형 Linux 이미지(약 5MB)로서 Docker 컨테이너의 기반으로 자주 사용됩니다. 이 문서에서는 구성된 IOx 디바이스, 즉 빈 CentOS 7 Linux 머신에서 시작하여 작은 Python 웹 서버를 구축하여 Docker 컨테이너에 패키징하고 이를 IOx 디바이스에 구축합니다.

구성

1. Linux 호스트에 IOx 클라이언트를 설치하고 준비합니다.

IOx 클라이언트는 애플리케이션을 패키지화하고 IOx 지원 장치와 통신하여 IOx 애플리케이션을 관리할 수 있는 툴입니다.

ioxclient 설치 패키지를 다운로드한 후 다음과 같이 설치할 수 있습니다.

```
[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
```

보시다시피, IOx 클라이언트의 첫 번째 실행에서 IOx 클라이언트로 관리할 수 있는 IOx 디바이스에 대한 프로파일을 생성할 수 있습니다. 나중에 이 작업을 수행하거나 설정을 추가/변경하려는 경우 나중에 이 명령을 실행할 수 있습니다. ioxclient 프로파일 만들기

2. Linux 호스트에 Docker를 설치하고 준비합니다.

Docker는 컨테이너를 구축하고 샘플 애플리케이션의 실행을 테스트하는 데 사용됩니다.

Docker를 설치하는 설치 단계는 설치하는 Linux OS에 따라 크게 달라집니다. 이 문서에서는 CentOS 7을 사용할 수 있습니다. 다양한 배포에 대한 설치 지침은 <https://docs.docker.com/engine/installation/>을 [참조하십시오](#).

설치 필수 구성 요소:

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

Docker repo를 추가합니다.

```
[jdepuydb ~]$ sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.
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
```

Docker 설치(설치 시 GPG 키 확인 수락):

```
[jdepuydb ~]$ sudo yum install docker-ce
...
Complete!
```

Docker 시작:

```
[jdepuydb ~]$ sudo systemctl start docker
```

```
[jdepuydb iox_docker_pythonweb]$ vi Dockerfile
[jdepuydb iox_docker_pythonweb]$ cat Dockerfile
FROM alpine:3.3
```

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

Docker를 일반 사용자로 액세스/실행하려면 이 사용자를 Docker 그룹에 추가하고 그룹 멤버십을 새로 고칩니다.

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

Docker Hub에 로그인합니다.

Docker Hub에는 사용할 수 있는 Alpine 기본 이미지가 포함되어 있습니다. 아직 Docker ID가 없는 경우 다음 사이트에 등록해야 합니다. <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
Username: jensdepuysdt
Password:
Login Succeeded
```

3. Python 웹 서버를 만듭니다.

이제 준비가 완료되었으므로 IOx 지원 디바이스에서 실행할 수 있는 실제 애플리케이션을 구축하기 시작할 수 있습니다.

```
[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()

```

이 코드는 webserver.py에서 만드는 매우 최소 Python 웹 서버입니다. 웹 서버는 GET이 요청되는 즉시 IOx python 웹 서버를 반환합니다. 웹 서버가 시작되는 포트는 포트 80 또는 webserver.py에 지정된 첫 번째 인수일 수 있습니다.

이 코드에는 또한 실행 함수에 로그 파일에 대한 쓰기가 포함되어 있습니다. 로그 파일은 IOx 클라이언트 또는 로컬 관리자로부터 상담을 받을 수 있습니다.

4. Dockerfile 및 Docker 컨테이너를 생성합니다.

이제 컨테이너에서 실행해야 하는 응용 프로그램(webserver.py)이 있으므로 Docker 컨테이너를 빌드해야 합니다. 컨테이너는 Dockerfile에 정의됩니다.

```

[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

```

보시다시피 Dockerfile도 간단하게 유지됩니다. Alpine 기본 이미지로 시작하여 Python을 설치하고 컨테이너의 루트에 webserver.py를 복사합니다.

Dockerfile이 준비되면 Docker 컨테이너를 빌드할 수 있습니다.

```

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

Docker build 명령은 Docker 파일에서 요청한 대로 기본 이미지를 다운로드하고 Python 및 종속성을 설치합니다. 마지막 명령은 확인을 위한 것입니다.

5. 생성된 Docker 컨테이너를 테스트합니다.

이 단계는 선택 사항이지만 방금 만든 Docker 컨테이너가 예상대로 작동할 준비가 되었는지 확인하는 것이 좋습니다.

```

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

/ # netstat -tlnp
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

```

netstat의 출력에서 볼 수 있듯이 webserver.py를 시작하면 포트 9000에서 수신 대기합니다.

6. Docker 컨테이너를 사용하여 IOx 패키지를 만듭니다.

컨테이너에서 웹 서버의 기능을 확인했으므로 이제 구축을 위한 IOx 패키지를 준비하고 빌드할 차례입니다. Dockerfile은 Docker 컨테이너를 빌드하는 지침을 제공하므로 package.yaml은 IOx 클라이언트에서 IOx 패키지를 빌드하는 지침을 제공합니다.

```

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"]

```

package.yaml의 내용에 대한 자세한 내용은 https://developer.cisco.com/media/iox-dev-guide-3-10-16/concepts/package_descriptor/을 [참조하십시오](#).

package.yaml 를 만든 후 IOx 패키지 작성을 시작할 수 있습니다.

첫 번째 단계는 Docker 이미지의 루트 FS를 내보내는 것입니다.

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

그런 다음 package.tar를 빌드할 수 있습니다.

```

[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 definition
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
```

빌드 결과는 IOx에 배포할 준비가 된 Docker 컨테이너가 포함된 IOx 패키지(package.tar)입니다.

 참고: IOxclient는 docker save 명령도 한 번에 수행할 수 있습니다. CentOS에서는 이 결과가 rootfs.tar 대신 기본 rootfs.img로 보내지기되므로 이 과정에서 나중에 문제가 발생합니다. 만
들 한 단계는 IOx 클라이언트 docker 패키지 IOxpythonweb:1.0을 사용하여 수행할 수 있습니
다.

8. IOx 디바이스에서 패키지를 배포, 활성화 및 시작합니다.

마지막 단계는 IOx 패키지를 IOx 디바이스에 구축하고 활성화한 후 시작하는 것입니다. 이러한 단
계는 IOx 클라이언트, 로컬 관리자 또는 Fog Network Director를 사용하여 수행할 수 있습니다. 이
문서에서는 IOx 클라이언트를 사용할 수 있습니다.

IOx 디바이스에 패키지를 구축하려면 이름 python_web을 사용합니다.

```
[jedepuyd@db iox_docker_pythonweb]$ ../ioxclient_1.3.0.0_linux_amd64/ioxclient app install python_web p
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_
Successfully deployed
```

애플리케이션을 활성화하기 전에 네트워크 컨피그레이션 방식을 정의해야 합니다. 그러기 위해서
는 JSON 파일을 생성해야 합니다. 활성화하면 활성화 요청에 첨부할 수 있습니다.

```
[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"}},
  }
}
[jedepuyd@db iox_docker_pythonweb]$ ../ioxclient_1.3.0.0_linux_amd64/ioxclient app activate python_web
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

마지막 작업은 방금 배포하고 활성화한 응용 프로그램을 시작하는 것입니다.

```
[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
```

포트 9000에서 HTTP 요청을 수신하도록 IOx 애플리케이션을 구성했으므로 컨테이너가 NAT 뒤에 있으므로 해당 포트를 IOx 디바이스에서 컨테이너로 전달해야 합니다. 이를 위해 Cisco IOS®에서 이 작업을 수행합니다.

```
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
```

첫 번째 명령은 GOS의 내부 IP 주소(IOx 컨테이너 시작/중지/실행)를 나열합니다.

두 번째 명령은 IOS 측의 Gi0 인터페이스에서 GOS에 대한 포트 9000에 대한 고정 포트 전달을 구성합니다. 디바이스가 L2 포트를 통해 연결된 경우(IR829의 경우) ip nat outside 명령문이 구성된 올바른 VLAN으로 Gi0 인터페이스를 교체해야 합니다.

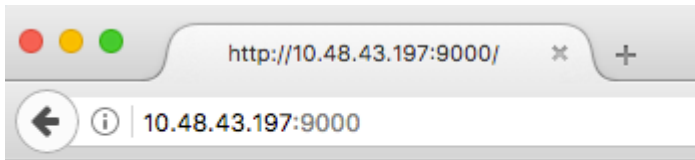
다음을 확인합니다.

구성이 올바르게 작동하는지 확인하려면 이 섹션을 활용하십시오.

웹 서버가 실행되고 올바르게 응답하는지 확인하려면 이 명령을 사용하여 웹 서버에 액세스하려고 시도할 수 있습니다.

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

또는 이미지에 표시된 실제 브라우저에서 사용할 수도 있습니다.

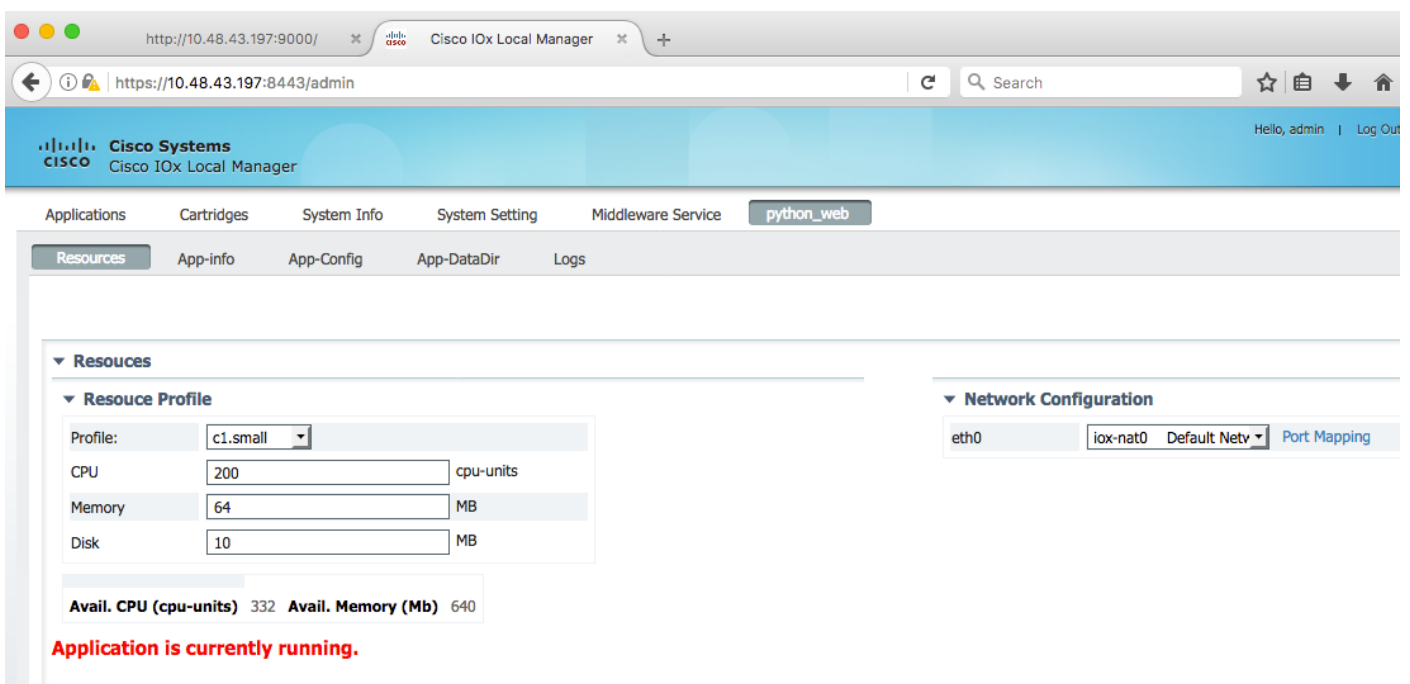


IOX python webserver

IOxclient CLI에서 애플리케이션 상태를 확인할 수도 있습니다.

```
[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
```

또한 이미지에 표시된 대로 로컬 관리자 GUI에서 애플리케이션 상태를 확인할 수도 있습니다.



webserver.py에서 쓰는 로그 파일을 보려면 다음을 수행합니다.

```
[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

문제 해결

이 섹션에서는 설정 문제 해결을 위해 사용할 수 있는 정보를 제공합니다.

응용 프로그램 및/또는 컨테이너의 문제를 해결하려면 다음을 실행하는 응용 프로그램의 콘솔에 연결하는 것이 가장 쉽습니다.

```
[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 -tln
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
```

이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.