Konfigurieren eines kleinen Alpine Linux Docker-Image auf IOx

Inhalt

Einführung

Dieses Dokument beschreibt den Konfigurationsprozess für die Erstellung, Bereitstellung und Verwaltung von Docker-basierten Anwendungen auf IOx-fähigen Cisco Geräten.

Voraussetzungen

Anforderungen

Für dieses Dokument bestehen keine speziellen Anforderungen.

Verwendete Komponenten

Die Informationen in diesem Dokument basieren auf den folgenden Software- und Hardwareversionen:

- IOx-fähiges Gerät, das für IOx konfiguriert ist:
  IP-Adresse konfiguriertGast-Betriebssystem (GOS) und Cisco Application Framework (CAF) werden ausgeführtNetwork Address Translation (NAT), konfiguriert für den Zugriff auf CAF (Port 8443)NAT für den Zugriff auf die GOS-Shell konfiguriert (Port 222)
- Linux-Host (für diesen Artikel wird eine minimale CentOS 7-Installation verwendet)
- IOx-Client-Installationsdateien, die unter folgender Adresse heruntergeladen werden können: https://software.cisco.com/download/release.html?mdfid=286306005&softwareid=286306762

Die Informationen in diesem Dokument wurden von den Geräten in einer bestimmten Laborumgebung erstellt. Alle in diesem Dokument verwendeten Geräte haben mit einer leeren (Standard-)Konfiguration begonnen. Wenn Ihr Netzwerk in Betrieb ist, stellen Sie sicher, dass Sie die potenziellen Auswirkungen eines Befehls verstehen.

Hintergrundinformationen

IOx kann verschiedene Pakettypen hosten, hauptsächlich Java, Python, LXC, Virtual Machine (VM) usw. Außerdem können Docker-Container ausgeführt werden. Cisco bietet ein Basisabbild und ein vollständiges Docker-Hub-Repository: https://devhub.cisco.com/artifactory/webapp/#/artifacts/browse/tree/General/iox-docker, das zum Erstellen von Docker-Containern verwendet werden kann.

Hier finden Sie eine schrittweise Anleitung zum Erstellen eines einfachen Docker-Containers unter Alpine Linux. Alpine Linux ist ein kleines Linux-Image (ca. 5 MB), das häufig als Basis für Docker-Container verwendet wird. In diesem Artikel starten Sie von einem konfigurierten IOx-Gerät, einem leeren CentOS 7 Linux-Rechner, und erstellen einen kleinen Python-Webserver, packen ihn in einen Docker-Container und stellen ihn auf einem IOx-Gerät bereit.

Konfigurieren

1. Installieren und Vorbereiten des IOx-Clients auf dem Linux-Host

Der IOx-Client ist das Tool, mit dem Anwendungen gepackt und mit dem IOx-fähigen Gerät kommuniziert werden können, um IOx-Anwendungen zu verwalten.

Nachdem Sie das ioxclient-Installationspaket heruntergeladen haben, kann es wie folgt installiert werden:

```
[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
```

Wie Sie sehen, kann beim ersten Start des IOx-Clients ein Profil für das IOx-Gerät erstellt werden, das Sie mit dem IOx-Client verwalten können. Falls Sie dies später tun möchten oder die Einstellungen hinzufügen/ändern möchten, können Sie diesen Befehl später ausführen: **ioxclient-Profile erstellen**

2. Installieren und Vorbereiten von Docker auf dem Linux-Host

Docker wird verwendet, um einen Container zu erstellen und die Ausführung unserer Beispielanwendung zu testen.

Die Installationsschritte zur Installation von Docker hängen stark vom Linux-Betriebssystem ab, unter dem Sie Docker installieren. In diesem Artikel können Sie CentOS 7 verwenden. Installationsanweisungen für verschiedene Distributionen finden Sie unter: https://docs.docker.com/engine/installation/.

Voraussetzungen installieren:

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

Fügen Sie den Docker-Report hinzu:

```
[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
```

Docker installieren (GPG-Schlüsselverifizierung bei der Installation akzeptieren):

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

Docker starten:

```
[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
```

Um Docker als regulären Benutzer aufrufen/ausführen zu können, fügen Sie diesen Benutzer der Docker-Gruppe hinzu, und aktualisieren Sie die Gruppenmitgliedschaft:

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

Beim Docker Hub anmelden:

Docker Hub enthält das Alpine-Basisbild, das Sie verwenden können. Falls Sie noch keine Docker-ID haben, müssen Sie sich registrieren auf: 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. Erstellen Sie den Python-Webserver.

Nachdem die Vorbereitung abgeschlossen ist, können Sie mit der Erstellung der eigentlichen Anwendung beginnen, die auf dem IOx-fähigen Gerät ausgeführt werden kann.

```
[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()
```

Dieser Code ist ein sehr minimaler Python-Webserver, den Sie in webserver.py erstellen. Der Webserver gibt einfach IOx python Webserver zurück, sobald GET angefordert wird. Der Port, auf dem der Webserver startet, kann entweder Port 80 oder das erste Argument sein, das webserver.py gegeben wurde.

Dieser Code enthält in der Laufzeit auch eine Schreibweise in eine Protokolldatei. Die Protokolldatei kann vom IOx-Client oder lokalen Manager abfragt werden.

4. Erstellen Sie den Dockerfile- und Docker-Container.

Nachdem Sie nun die Anwendung (webserver.py) haben, die in Ihrem Container ausgeführt werden soll, ist es an der Zeit, den Docker-Container zu erstellen. Ein Container ist in einer

Dockerfile definiert:

```
[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
```

Wie Sie sehen, wird die Dockerfile auch einfach gehalten. Sie beginnen mit dem Alpine-Basisbild, installieren Python und kopieren Ihre webserver.py in das Stammverzeichnis des Containers.

Sobald Sie Ihre Dockerfile fertig haben, können Sie den Docker-Container erstellen:

```
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
```

Der Docker-Buildbefehl lädt das Basisimage herunter und installiert Python und Abhängigkeiten, wie in der Dockerfile angefordert. Der letzte Befehl dient zur Überprüfung.

5. Testen Sie den erstellten Docker-Container.

Dieser Schritt ist optional, aber es ist gut zu überprüfen, ob der gerade erstellte Docker-Container wie erwartet funktioniert.

```
[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
```

Wie Sie in der Ausgabe von netstat sehen können, hört der Webserver.py nach dem Start auf Port 9000 zu.

6. Erstellen Sie das IOx-Paket mit dem Docker-Container.

Nachdem Sie die Funktionalität Ihres Webservers im Container überprüft haben, ist es an der Zeit, das IOx-Paket für die Bereitstellung vorzubereiten und zu erstellen. Da die Dockerfile Anweisungen zum Erstellen eines Docker-Containers bereitstellt, enthält package.yaml Anweisungen für den IOx-Client zum Erstellen Ihres IOx-Pakets.

```
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"]
```

Weitere Informationen zum Inhalt der Datei package.yaml finden Sie hier: https://developer.cisco.com/media/iox-dev-guide-3-10-16/concepts/package_descriptor/.

Nachdem Sie die Datei package.yaml erstellt haben, können Sie mit der Erstellung des IOx-Pakets beginnen.

Der erste Schritt besteht darin, die Root-FS des Docker-Images zu exportieren:

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

Als Nächstes können Sie die Datei package.tar erstellen:

```
[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
```

Das Ergebnis des Builds ist ein IOx-Paket (package.tar), das den Docker-Container enthält, der für die Bereitstellung auf IOx bereit ist.

> **Hinweis**: IOxclient kann auch den Befehl docker save in einem Schritt ausführen. Bei CentOS wird dadurch der Export in die Standard-Rootfs.img anstelle von rootfs.tar durchgeführt, was später Schwierigkeiten verursacht. Der eine Schritt zum Erstellen ist mit der Verwendung von IOx Client Docker Paket IOxpythonweb:1.0 möglich.

8. Bereitstellen, aktivieren und starten Sie das Paket auf dem IOx-Gerät.

Die letzten Schritte sind die Bereitstellung des IOx-Pakets auf dem IOx-Gerät, die Aktivierung und das Starten. Diese Schritte können entweder mit dem IOx-Client, dem Local Manager oder Fog Network Director ausgeführt werden. In diesem Artikel können Sie den IOx-Client verwenden.

Um das Paket auf dem IOx-Gerät bereitzustellen, verwenden Sie den Namen 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
```

Bevor Sie die Anwendung aktivieren können, müssen Sie definieren, wie die Netzwerkkonfiguration aussehen soll. Dazu müssen Sie eine JSON-Datei erstellen. Wenn Sie die Aktivierung durchführen, kann sie an die Aktivierungsanfrage angeschlossen werden.

```
[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
```

Die letzte Aktion besteht darin, die Anwendung zu starten, die Sie gerade bereitgestellt und aktiviert haben:

```
[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
```

Da Sie die IOx-Anwendung so konfiguriert haben, dass sie Port 9000 auf HTTP-Anfragen überwacht, müssen Sie diesen Port vom IOx-Gerät an den Container weiterleiten, da sich der Container hinter NAT befindet. Führen Sie dies auf Cisco IOS® aus, um dies zu tun.

```
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
```

Der erste Befehl listet die interne IP-Adresse von GOS auf (verantwortlich für Start/Stopp/Ausführen der IOx-Container).

Der zweite Befehl konfiguriert einen statischen Port Forward für Port 9000 an der Gi0-Schnittstelle der IOS-Seite an GOS. Falls Ihr Gerät über einen L2-Port angeschlossen ist (was bei IR829 höchstwahrscheinlich der Fall ist), müssen Sie die Gi0-Schnittstelle durch das richtige VLAN ersetzen, für das die externe IP-Adresse konfiguriert ist.

Überprüfen

In diesem Abschnitt überprüfen Sie, ob Ihre Konfiguration ordnungsgemäß funktioniert.

Um zu überprüfen, ob der Webserver ordnungsgemäß ausgeführt wird und reagiert, können Sie versuchen, mit diesem Befehl auf den Webserver zuzugreifen.

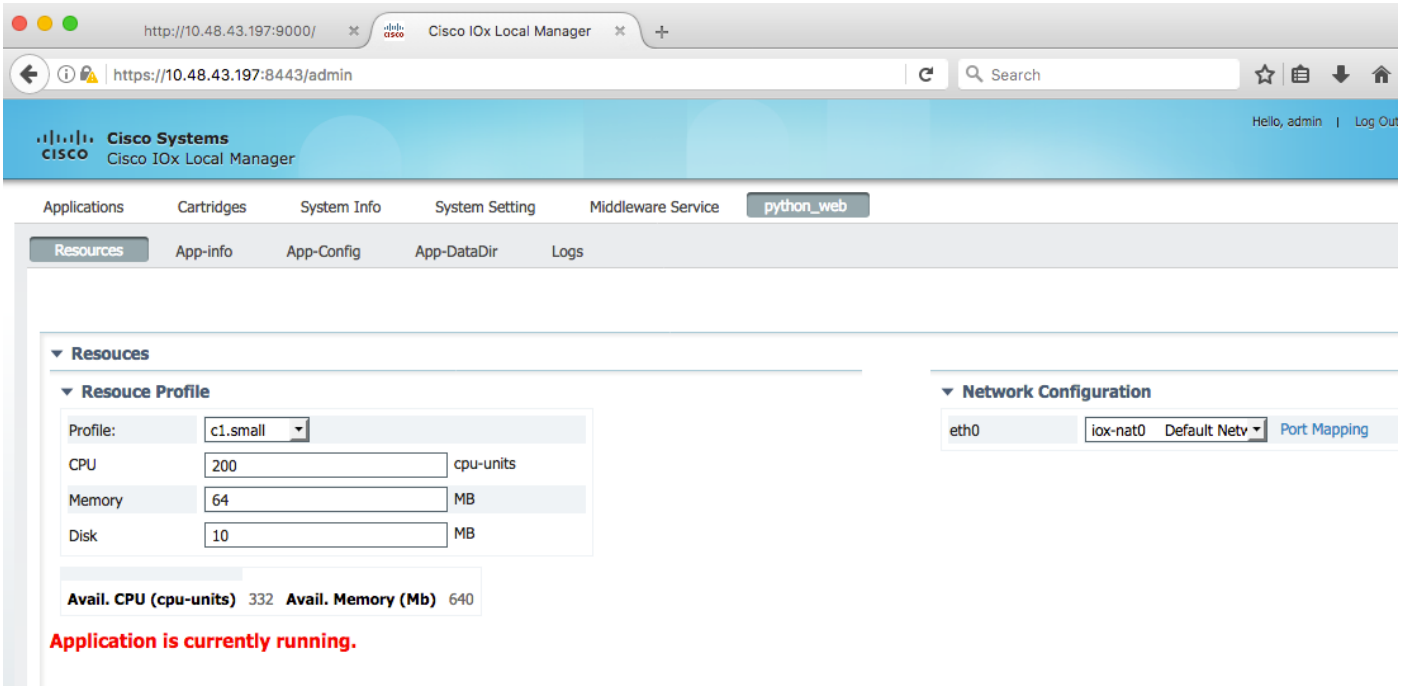
```
[jedepuyd@db iox_docker_pythonweb]$ curl http://10.48.43.197:9000/
<html><body><h1>IOX python webserver</h1></body></html>
```

Oder von einem echten Browser, wie im Bild gezeigt.

Sie können den Anwendungsstatus auch über die IOxclient-CLI überprüfen:

```
[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
```

und Sie können den Anwendungsstatus auch über die Benutzeroberfläche des lokalen Managers überprüfen, wie im Bild gezeigt.



Um einen Blick auf die Protokolldatei zu werfen, auf die Sie in webserver.py schreiben:

```
[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
```

Fehlerbehebung

Dieser Abschnitt enthält Informationen, die Sie zur Fehlerbehebung bei Ihrer Konfiguration verwenden können.

Um eine Fehlerbehebung für die Anwendung und/oder den Container durchzuführen, können Sie am einfachsten eine Verbindung zur Konsole der Anwendung herstellen, die ausgeführt wird:

```
[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
```