IOxでのSmall Alpine Linux Dockerイメージの設定

内容

概要

このドキュメントでは、Cisco IOx対応デバイス上でDockerベースのアプリケーションを作成、導入、および管理するための設定プロセスについて説明します。

前提条件

要件

このドキュメントに特有の要件はありません。

使用するコンポーネント

このドキュメントの情報は、次のソフトウェアとハードウェアのバージョンに基づいています。

- IOx対応のデバイス。IOx用に設定されます。
  設定されたIPアドレスゲストオペレーティングシステム(GOS)およびCisco Application Framework(CAF)の実行CAF（ポート8443）へのアクセス用に設定されたネットワークアドレス変換(NAT)GOSシェル（ポート2222）へのアクセス用に設定されたNAT
- Linuxホスト（この記事では最小限のCentOS 7インストールを使用）
- IOxクライアントのインストールファイル（ダウンロード可能）:https://software.cisco.com/download/release.html?mdfid=286306005&softwareid=286306762

このドキュメントの情報は、特定のラボ環境にあるデバイスに基づいて作成されました。このドキュメントで使用するすべてのデバイスは、初期（デフォルト）設定の状態から起動しています。対象のネットワークが実稼働中である場合には、どのようなコマンドについても、その潜在的な影響について確実に理解しておく必要があります。

背景説明

IOxは、主にJava、Python、LXC、仮想マシン(VM)などの異なるタイプのパッケージをホストでき、またDockerコンテナを実行することもできます。シスコは、Dockerコンテナの構築に使用できる基本イメージと完全なDockerハブリポジトリ https://devhub.cisco.com/artifactory/webapp/#/artifacts/browse/tree/General/iox-dockerを提供しています。

ここでは、Alpine Linuxを使用して簡単なDockerコンテナを構築する方法をステップバイステップで説明します。Alpine Linuxは小さなLinuxイメージ（約5MB）で、Dockerコンテナのベースとしてよく使用されます。この記事では、設定されたIOxデバイス、空のCentOS 7 Linuxマシンから始めて、小さなPython Webサーバを構築し、それをDockerコンテナにパッケージ化して、IOxデバイスに展開します。

設定

1. LinuxホストにIOxクライアントをインストールして準備する。

IOxクライアントは、アプリケーションをパッケージ化し、IOx対応デバイスと通信してIOxアプリケーションを管理できるツールです。

ioxclientインストールパッケージをダウンロードした後、次のようにインストールできます。

```
[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
```

ご覧のように、IOxクライアントの最初の起動時に、IOxクライアントで管理できるIOxデバイスのプロファイルを生成できます。後で実行する場合、または設定を追加または変更する場合は、後で次のコマンドを実行できます。**ioxclient profiles create**

2. LinuxホストにDockerをインストールして準備します。

Dockerは、コンテナを構築し、サンプルアプリケーションの実行をテストするために使用されます。

Dockerのインストール手順は、インストール先のLinux OSによって大きく異なります。この記事では、CentOS 7を使用できます。さまざまなディストリビューションのインストール手順については、https://docs.docker.com/engine/installation/を参照してください。

インストールの前提条件：

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

Docker repoを追加します。

```
[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をインストールします（インストール時にGPGキーの確認を受け入れます）。

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

Dockerの起動：

```
[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
```

通常のユーザとしてDockerにアクセス/実行できるようにするには、このユーザをDockerグループに追加し、グループメンバーシップを更新します。

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

Docker Hubにログインします。

Docker Hubには、使用できるAlpineベースイメージが含まれています。Docker IDをまだ持っていない場合は、次のページに登録する必要があります。https://hub.docker.com/ にアクセスしてください。

```
[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. Python Webサーバを作成します。

準備が完了したら、IOx対応デバイスで実行できる実際のアプリケーションの構築を開始できます。

```
[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()
```

このコードは、webserver.pyで作成する最小限のPython Webサーバです。Webサーバは、GETが要求されるとすぐにIOx python webserverを返すだけです。Webサーバが起動するポートは、ポート80またはwebserver.pyに与えられた最初の引数のいずれかです。

このコードには、run関数でログファイルへの書き込みも含まれます。ログファイルは、IOxクライアントまたはローカルマネージャから相談できます。

4. DockerfileとDockerコンテナを作成します。

コンテナ内で実行するアプリケーション(webserver.py)が完成したので、Dockerコンテナを構築します。コンテナはDockerfileで定義されます。

```
[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
```

ご覧のように、Dockerfileもシンプルに保たれています。まずAlpineベースイメージから始めて、Pythonをインストールし、webserver.pyをコンテナのルートにコピーします。

Dockerfileの準備ができたら、Dockerコンテナを構築できます。

```
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コマンドはベースイメージをダウンロードし、Dockerfileで要求されたとおりにPythonと依存関係をインストールします。最後のコマンドは検証用です。

5.作成したDockerコンテナをテストします。

この手順はオプションですが、作成したばかりのDockerコンテナが期待どおりに動作する準備ができていることを確認することをお勧めします。

```
[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
```

netstatの出力からわかるように、webserver.pyを起動した後、ポート9000でリッスンします。

6. DockerコンテナでIOxパッケージを作成します。

コンテナ内のWebサーバの機能を確認したら、IOxパッケージを準備して構築します。DockerfileはDockerコンテナを構築する手順を提供するため、package.yamlはIOxクライアントがIOxパッケージを構築する手順を提供します。

```
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.yamlの内容の詳細については、https://developer.cisco.com/media/iox-dev-guide-3-10-16/concepts/package_descriptor/をご覧ください。

package.yamlを作成したら、IOxパッケージの構築を開始できます。

最初のステップは、DockerイメージのルートFSをエクスポートすることです。

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

次に、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
```

ビルドの結果、IOxパッケージ(package.tar)が作成されます。このパッケージには、Dockerコンテナが含まれており、IOx上に展開できます。

> **注：**IOxclientはdocker saveコマンドを1つのステップで実行することもできます。CentOSでは、この結果をrootfs.tarではなくデフォルトのrootfs.imgにエクスポートするため、プロセスの後半で問題が発生します。作成する1つのステップは、IOx client docker package IOxpythonweb:1.0を使用して行うことができます。

8. IOxデバイスにパッケージを展開、アクティブ化、および起動します。

最後の手順は、IOxパッケージをIOxデバイスに展開し、アクティブ化して開始することです。これらの手順は、IOxクライアント、Local Manager、またはFog Network Directorを使用して実行できます。この記事では、IOxクライアントを使用できます。

パッケージをIOxデバイスに展開するには、python_webという名前を使用します。

```
[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
```

アプリケーションをアクティブにする前に、ネットワーク設定の方法を定義する必要があります。そのためには、JSONファイルを作成する必要があります。アクティブ化すると、アクティブ化要求に関連付けることができます。

```
[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
```

最後のアクションは、デプロイしてアクティブにしたアプリケーションを起動することです。

```
[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
```

ポート9000でtor HTTP要求をリッスンするようにIOxアプリケーションを設定しているため、コンテナがNATの背後にあるため、IOxデバイスからそのポートをコンテナに転送する必要があります。Cisco IOS®でこれを実行します。

```
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
```

最初のコマンドは、GOSの内部IPアドレスをリストします（IOxコンテナの開始/停止/実行を担当）。

2番目のコマンドは、IOS側のGi0インターフェイスのポート9000のスタティックポート転送をGOSに設定します。デバイスがL2ポート（IR829の場合が最も多い）経由で接続されている場合は、Gi0インターフェイスを、ip nat outside文が設定されている正しいVLANに置き換える必要があります。

確認

ここでは、設定が正常に機能しているかどうかを確認します。

Webサーバが稼働していて、正しく応答しているかどうかを確認するには、次のコマンドを使用してWebサーバへのアクセスを試すことができます。

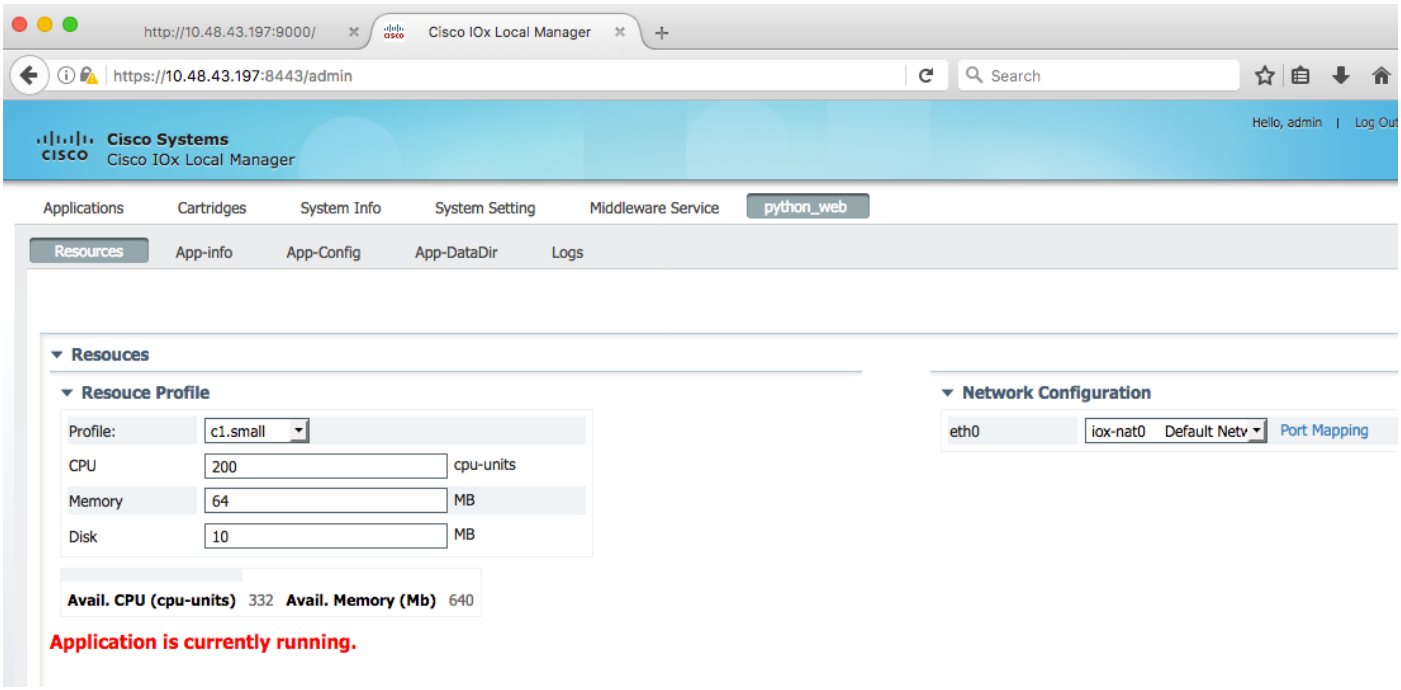
```
[jedepuyd@db iox_docker_pythonweb]$ curl http://10.48.43.197:9000/
<html><body><h1>IOX python webserver</h1></body></html>
```

または、図に示すように、実際のブラウザから実行します。

IOxclient CLIからアプリケーションステータスを確認することもできます。

```
[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
```

また、図に示すように、Local Manager GUIからアプリケーションステータスを確認することもできます。



webserver.pyで書き込むログファイルを確認するには、次のコマンドを実行します。

```
[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
```

トラブルシュート

ここでは、設定のトラブルシューティングに使用できる情報を示します。

アプリケーションやコンテナのトラブルシューティングを行うには、最も簡単な方法は、次のアプリケーションを実行するコンソールに接続することです。

```
[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
```