Configurar uma pequena imagem do Linux Docker em IOx

Contents

Introduction

Este documento descreve o processo de configuração para criar, implantar e gerenciar aplicativos baseados em Docker em dispositivos compatíveis com Cisco IOx.

Prerequisites

Requirements

Não existem requisitos específicos para este documento.

Componentes Utilizados

As informações neste documento são baseadas nestas versões de software e hardware:

- Dispositivo compatível com IOx configurado para IOx:
  Endereço IP configuradoSistema operacional convidado (GOS) e Cisco Application Framework (CAF) em execuçãoNetwork Address Translation (NAT) configurada para acesso ao CAF (porta 8443)NAT configurado para acesso à shell GOS (porta 2222)
- Host Linux (uma instalação mínima do CentOS 7 é usada para este artigo)
- Arquivos de instalação do cliente IOx que podem ser baixados de:
  https://software.cisco.com/download/release.html?mdfid=286306005&softwareid=286306762

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Informações de Apoio

O IOx pode hospedar diferentes tipos de pacotes, principalmente Java, Python, LXC, Virtual Machine (VM) etc., e também pode executar contêineres Docker. A Cisco oferece uma imagem

básica e um repositório completo do hub Docker: https://devhub.cisco.com/artifactory/webapp/#/artifacts/browse/tree/General/iox-docker que pode ser usado para criar contêineres Docker.

Este é um guia passo a passo sobre como construir um simples contêiner Docker com o uso de Linux Alpino. O Linux Alpino é uma pequena imagem Linux (cerca de 5 MB), que é frequentemente usada como base para contêineres Docker. Neste artigo, você começa com um dispositivo IOSx configurado, uma máquina Linux CentOS 7 vazia e constrói um pequeno servidor Web Python, o empacota em um contêiner Docker e o implanta em um dispositivo IOx.

Configurar

1. Instalar e preparar o cliente IOx no host Linux.

O cliente IOx é a ferramenta que pode empacotar aplicativos e se comunicar com o dispositivo compatível com IOx para gerenciar aplicativos IOx.

Depois de baixar o pacote de instalação do ioxclient, ele pode ser instalado da seguinte maneira:

```
[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 você pode ver, na primeira inicialização do cliente IOx, um perfil pode ser gerado para o dispositivo IOx que você pode gerenciar com o cliente IOx. Caso deseje fazer isso mais tarde ou se quiser adicionar/alterar as configurações, execute este comando mais tarde: **perfis ioxclient criam**

2. Instale e prepare o Docker no host Linux.

O Docker é usado para construir um contêiner e testar a execução de nosso aplicativo de amostra.

As etapas de instalação para instalar o Docker dependem muito do sistema operacional Linux no qual você o instala. Para este artigo, você pode usar CentOS 7. Para obter instruções de instalação para distribuições diferentes, consulte: https://docs.docker.com/engine/installation/.

Instalar pré-requisitos:

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

Adicione o acordo de recompra do 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
```

Instale o Docker (aceite a verificação da chave GPG quando instalar):

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

Iniciar 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
```

Para poder acessar/executar o Docker como um usuário regular, adicione esse usuário ao grupo Docker e atualize a associação ao grupo:

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

Faça login no Docker Hub:

O Docker Hub contém a imagem base alpina que você pode usar. Caso ainda não tenha uma ID do Docker, você precisa se registrar em: 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. Crie o servidor Web Python.

Agora que a preparação está concluída, você pode começar a criar o aplicativo real que pode ser executado no dispositivo habilitado para 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()
```

Este código é um servidor Web Python muito mínimo, que você cria no webserver.py. O servidor Web simplesmente retorna o servidor web IOx python assim que um GET é solicitado. A porta na qual o servidor web é iniciado pode ser a porta 80 ou o primeiro argumento dado ao webserver.py.

Este código também contém, na função de execução, uma gravação em um arquivo de log. O arquivo de log está disponível para consulta do cliente IOx ou do gerente local.

4. Crie o arquivo Dockerfile e o contêiner Docker.

Agora que você tem o aplicativo (webserver.py) que deve ser executado em seu contêiner, é hora de criar o contêiner Docker. Um contêiner é definido em um arquivo 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 você pode ver, o arquivo Dockerfile também é simples. Você começa com a imagem base alpina, instala o Python e copia seu servidor web.py para a raiz do contêiner.

Depois de ter seu arquivo Dockerfile pronto, você pode criar o contêiner 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
```

O comando de compilação do Docker faz o download da imagem base e instala Python e dependências, conforme solicitado no arquivo Dockerfile. O último comando é para verificação.

5. Teste o contêiner Docker criado.

Esta etapa é opcional, mas é bom verificar se o contêiner do Docker recém-criado está pronto para funcionar conforme o esperado.

```
[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 você pode ver na saída do netstat, depois que você inicia o webserver.py, ele escuta na porta 9000.

6. Crie o pacote IOx com o contêiner Docker.

Agora que você verificou a funcionalidade do seu servidor Web no contêiner, é hora de preparar e criar o pacote IOx para implantação. Como o Dockerfile fornece instruções para criar um contêiner Docker, o package.yaml fornece instruções para que o cliente IOx crie seu pacote 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"]
```

Mais informações sobre o conteúdo do pacote.yaml podem ser encontradas aqui: https://developer.cisco.com/media/iox-dev-guide-3-10-16/concepts/package_descriptor/.

Depois de criar o pacote.yaml, você pode começar a criar o pacote IOx.

A primeira etapa é exportar o FS raiz da imagem do Docker:

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

Em seguida, você pode criar o pacote.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
```

O resultado da compilação é um pacote IOx (package.tar), que contém o contêiner Docker, pronto para ser implantado em IOx.

> **Note**: O IOxclient também pode executar o comando docker save em uma etapa. No CentOS, isso resulta na exportação para o rootfs.img padrão em vez de rootfs.tar, o que gera problemas posteriormente no processo. A única etapa a ser criada pode ser feita com o uso do pacote do docker do cliente IOx IOxpythonweb:1.0.

8. Implante, ative e inicie o pacote no dispositivo IOx.

As últimas etapas são implantar o pacote IOx no dispositivo IOx, ativá-lo e iniciar. Essas etapas podem ser feitas com o uso de cliente IOx, Gerenciador local ou Fog Network Diretor. Para este artigo, você pode usar o cliente IOx.

Para implantar o pacote no dispositivo IOx, use o 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
```

Antes de ativar o aplicativo, você precisa definir como seria a configuração da rede. Para fazer isso, você precisa criar um arquivo JSON. Quando a ativação é efetuada, pode ser anexada à solicitação de ativação.

```
[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
```

A última ação aqui é iniciar o aplicativo que você acabou de implantar e ativar:

```
[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
```

Como você configurou seu aplicativo IOx para escutar na porta 9000 para solicitações HTTP do operador, ainda é necessário encaminhar essa porta do dispositivo IOx para o contêiner, pois o contêiner está atrás do NAT. Faça isso no 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
```

O primeiro comando lista o endereço IP interno do GOS (responsável por iniciar/parar/executar os contêineres IOx).

O segundo comando configura uma porta estática encaminhada para a porta 9000 na interface Gi0 do lado do IOS para o GOS. Caso seu dispositivo esteja conectado por uma porta L2 (o que é provavelmente o caso em IR829), você precisará substituir a interface Gi0 pela VLAN correta, que tem a instrução ip nat outside configurada.

Verificar

Use esta seção para confirmar se a sua configuração funciona corretamente.

Para verificar se o servidor web é executado e responde corretamente, você pode tentar acessar o servidor web com esse comando.

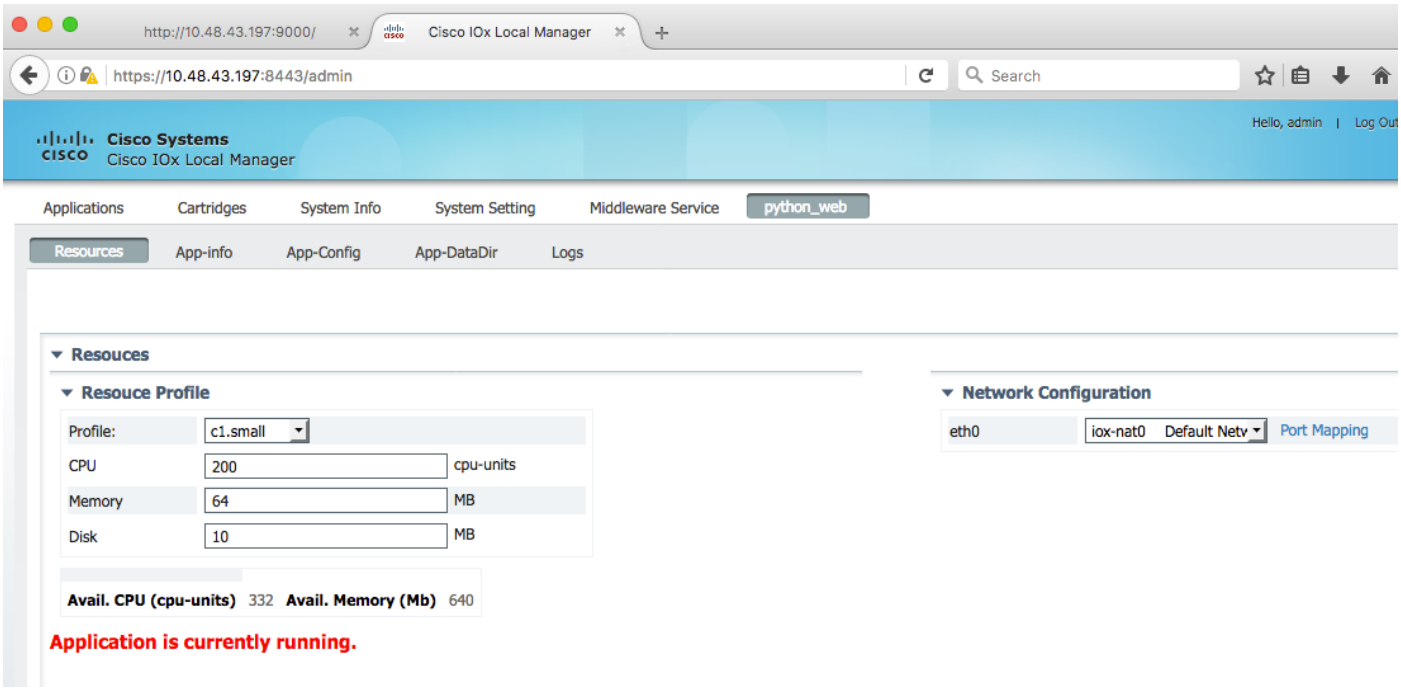
```
[jedepuyd@db iox_docker_pythonweb]$ curl http://10.48.43.197:9000/
<html><body><h1>IOX python webserver</h1></body></html>
```

Ou de um navegador real, como mostrado na imagem.

Você também pode verificar o status do aplicativo na CLI do 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
```

e você também pode verificar o status do aplicativo na GUI do Local Manager, como mostrado na imagem.



Para ver o arquivo de log no qual você grava no 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
```

Troubleshoot

Esta seção disponibiliza informações para a solução de problemas de configuração.

Para solucionar problemas do aplicativo e/ou contêiner, a maneira mais fácil é conectar-se ao console do aplicativo que executa:

```
[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
```