在IOx上配置小型Alpine Linux Docker映像

目錄

簡介

本文檔介紹在支援Cisco IOx的裝置上建立、部署和管理基於Docker的應用程式的配置過程。

必要條件

需求

本文件沒有特定需求。

採用元件

本文中的資訊係根據以下軟體和硬體版本：

- 為IOx配置的支援IOx的裝置：
  已配置IP地址訪客作業系統(GOS)和思科應用架構(CAF)正在運行網路地址轉換(NAT)配置為訪問CAF（埠8443）NAT配置為訪問GOS外殼（埠2222）
- Linux主機（本文使用最小的CentOS 7安裝）
- IOx客戶端安裝檔案，可從以下位置下載：https://software.cisco.com/download/release.html?mdfid=286306005&softwareid=286306762

本文中的資訊是根據特定實驗室環境內的裝置所建立。文中使用到的所有裝置皆從已清除（預設）的組態來啟動。如果您的網路正在作用，請確保您已瞭解任何指令可能造成的影響。

背景資訊

IOx可以託管不同型別的包，主要是Java、Python、LXC、虛擬機(VM)等，還可以運行Docker容器。思科提供一個基本映像和一個完整的Docker中心儲存庫：https://devhub.cisco.com/artifactory/webapp/#/artifacts/browse/tree/General/iox-docker，可用於構建Docker容器。

以下分步指南介紹了如何使用Alpine Linux構建一個簡單的Docker容器。Alpine Linux是一個小型

Linux映像（約5MB），常被用作Docker容器的基礎。在本文中，您從配置的IOx裝置（一個空的CentOS 7 Linux電腦）開始，然後構建一個小型Python Web伺服器，將其打包到Docker容器中並部署到IOx裝置上。

設定

1.在Linux主機上安裝和準備IOx客戶端。

IOx客戶端是一種工具，可以打包應用程式並與支援IOx的裝置進行通訊以管理IOx應用程式。

下載ioxclient安裝軟體包後，可以按如下方式安裝：

```
[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
```

如您所見，在第一次啟動IOx客戶端時，可以為IOx裝置生成一個配置檔案，您可以通過IOx客戶端對其進行管理。如果您想以後再執行此操作或想要新增/更改設定，可以以後運行此命令：**ioxclient profiles create**

2.在Linux主機上安裝和準備Docker。

Docker用於構建容器並測試示例應用程式的執行。

安裝Docker的安裝步驟在很大程度上取決於安裝它的Linux作業系統。本文可以使用CentOS 7。有關不同發行版的安裝說明，請參閱https://docs.docker.com/engine/installation/。

安裝前提條件：

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

新增Docker回購：

```
[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
```

安裝Docker（安裝時接受GPG金鑰驗證）：

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

啟動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
```

為了能夠作為常規使用者訪問/運行Docker，請將此使用者新增到Docker組並刷新組成員身份：

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

登入到Docker Hub:

Docker中心包含可供使用的阿爾卑斯山基本映像。如果您還沒有Docker ID，則需要在以下位置註冊：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.建立Python Web伺服器。

完成準備後，您可以開始構建可在啟用IOx的裝置上運行的實際應用程式。

```
[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()
```

此代碼是一個非常最小的Python Web伺服器，您可在webserver.py中建立它。Web伺服器只需在請求GET時立即返回IOx python webserver。Web伺服器啟動的埠可以是埠80，也可以是webserver.py的第一個引數。

在run函式中，此代碼還包含對日誌檔案的寫入。日誌檔案可從IOx客戶端或本地管理器中查閱。

4.建立Dockerfile和Docker容器。

既然您已經擁有應該在容器中運行的應用程式(webserver.py)，那麼是時候構建Docker容器了。容器在Dockerfile中定義：

```
[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
```

您可以看到，Dockerfile也保持簡單。從Alpine base映像開始，安裝Python並將webserver.py複製到容器的根目錄。

準備好Dockerfile後，可以生成Docker容器：

```
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命令可下載基本映像，並根據您在Dockerfile中的請求安裝Python和依賴項。最後一個命令是進行驗證。

5.測試建立的Docker容器。

此步驟是可選的，但最好驗證您剛剛構建的Docker容器是否可以如預期那樣工作。

```
[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
```

在netstat的輸出中可看到，啟動webserver.py後，它會偵聽埠9000。

6.使用Docker容器建立IOx包。

現在，您已驗證容器中Web伺服器的功能，現在應準備並構建IOx包進行部署。由於Dockerfile提供了構建Docker容器的說明，package.yaml提供了IOx客戶端構建IOx容器的說明。

```
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"]
```

有關package.yaml內容的詳細資訊，請訪問：https://developer.cisco.com/media/iox-dev-guide-3-10-16/concepts/package_descriptor/。

建立package.yaml後，可以開始生成IOx程式包。

第一步是匯出Docker映像的根FS:

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

接下來，您可以生成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
```

生成的結果是一個IOx包(package.tar)，它包含Docker容器，準備部署在IOx上。

附註：IOxclient也可以一步完成docker save命令。在CentOS上，這會導致匯出到預設的rootfs.img而不是rootfs.tar，後者會在流程的後續階段產生問題。只需使用IOx客戶端docker軟體包IOxpythonweb:1.0即可完成建立步驟。

8.在IOx裝置上部署、啟用和啟動包。

最後一步是將IOx包部署到IOx裝置，啟用後啟動。這些步驟可以使用IOx客戶端、本地管理器或霧網路控制器來完成。對於本文，您可以使用IOx客戶端。

要將軟體包部署到IOx裝置，請使用名稱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
```

在啟用應用程式之前，您需要定義網路配置的方式。為此，您需要建立JSON檔案。啟用後，可以將其附加到啟用請求。

```
[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
```

這裡的最後一個操作是啟動您剛剛部署和啟用的應用程式：

```
[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
```

由於您已將IOx應用程式配置為在埠9000上偵聽或HTTP請求，因此仍需要將該埠從IOx裝置轉發到容器，因為容器位於NAT之後。在Cisco IOS®上執行該操作即可。

```
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
```

第一個命令列出GOS的內部IP地址（負責啟動/停止/運行IOx容器）。

第二個命令將IOS端Gi0介面上的埠9000的靜態埠轉發配置到GOS。如果您的裝置通過L2埠連線（很可能是IR829上的情況），則需要用配置了ip nat outside語句的正確VLAN替換Gi0介面。

驗證

使用本節內容，確認您的組態是否正常運作。

若要確認Web伺服器是否正常執行且正確回應，您可以嘗試使用以下命令存取Web伺服器。

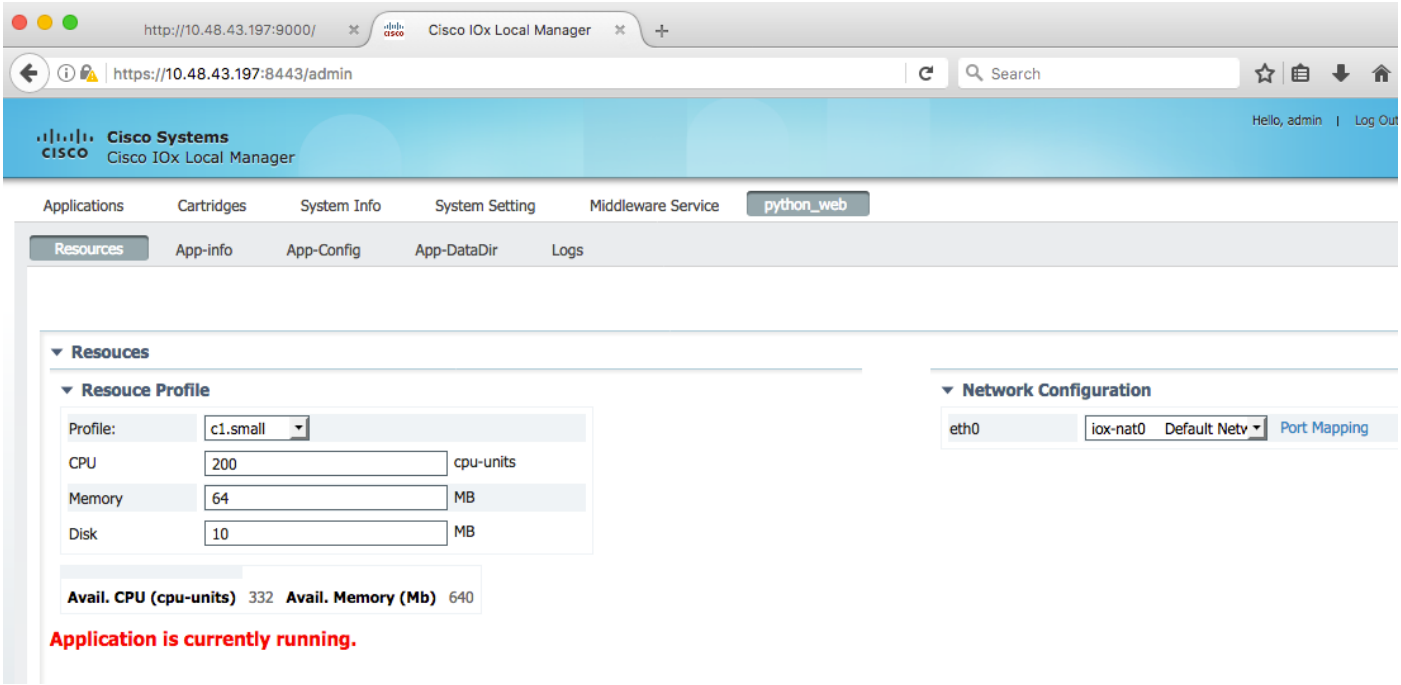
```
[jedepuyd@db iox_docker_pythonweb]$ curl http://10.48.43.197:9000/
<html><body><h1>IOX python webserver</h1></body></html>
```

或者，從實際瀏覽器中（如圖所示）。

您還可以通過IOxclient CLI驗證應用程式狀態：

```
[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
```

您還可以從本地管理器GUI驗證應用程式狀態，如下圖所示。



若要檢視您在webserver.py中寫入的日誌檔案：

```
[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
```

疑難排解

本節提供的資訊可用於對組態進行疑難排解。

為了對應用程式和/或容器進行故障排除，最簡單的方法是連線到運行的應用程式的控制檯：

```
[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
```