在IOx上配置小型Alpine Linux Docker映像

目录

简介

先决条件

- 要求
- 使用的组件

背景信息

配置

验证

故障排除

简介

本文档介绍在支持Cisco IOx的设备上创建、部署和管理基于Docker的应用的配置流程。

先决条件

要求

本文档没有任何特定的要求。

使用的组件

本文档中的信息基于以下软件和硬件版本：

- 为IOx配置的支持IOx的设备：
  - 已配置IP地址
  - 运行访客操作系统(GOS)和思科应用框架(CAF)
  - 网络地址转换(NAT)配置为访问CAF（端口8443）
  - NAT配置为访问GOS外壳（端口2222）
- Linux主机（本文使用的是最小的CentOS 7安装）
- 可从以下位置下载的IOx客户端安装文件：https://software.cisco.com/download/release.html?mdfid=286306005&softwareid=286306762

本文档中的信息都是基于特定实验室环境中的设备编写的。本文档中使用的所有设备最初均采用原始（默认）配置。如果您使用的是真实网络，请确保您已经了解所有命令的潜在影响。

背景信息

IOx可以托管不同类型的包，主要是Java、Python、LXC、虚拟机(VM)等，还可以运行Docker容器。思科提供基础映像和完整的Docker中心存储库

：https://devhub.cisco.com/artifactory/webapp/#/artifacts/browse/tree/General/iox-docker，可用于构建Docker容器。

以下分步指南介绍如何使用Alpine Linux构建简单的Docker容器。Alpine Linux是一个小型Linux映像（约5MB），常被用作Docker容器的基础。在本文中，您将从已配置的IOx设备（一个空的CentOS 7 Linux计算机）开始，然后构建一个小型Python Web服务器，将其打包到Docker容器中，然后将其部署到IOx设备上。

配置

1.在Linux主机上安装和准备IOx客户端。

IOx客户端是一种工具，可以打包应用并与支持IOx的设备进行通信以管理IOx应用。

下载ioxclient安装软件包后，可以按如下方式安装：

```
[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
```

如您所见，在首次启动IOx客户端时，可以为IOx设备生成一个配置文件，您可以通过IOx客户端对其进行管理。如果您想稍后执行此操作，或者想添加/更改设置，可以稍后运行此命令：ioxclient profiles create

2.在Linux主机上安装和准备Docker。

Docker用于构建容器并测试示例应用程序的执行。

安装Docker的安装步骤在很大程度上取决于安装它的Linux操作系统。本文可以使用CentOS 7。有关不同发行版的安装说明，请参阅https://docs.docker.com/engine/installation/。

安装必备条件：

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

添加Docker回购：

```
[jedepuyd@db ~]$ 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
```

安装Docker（安装时接受GPG密钥验证）：

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

开始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
```

为了能够作为常规用户访问/运行Docker，请将此用户添加到Docker组并刷新组成员身份：

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

登录到Docker Hub:

Docker集线器包含可供使用的Alpine基本映像。如果您尚没有Docker ID，则需要在以下位置注册：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, hea
Username: jensdepuydt
Password:
Login Succeeded
```

3.创建Python Web服务器。

准备完成后，您可以开始构建可在启用IOx的设备上运行的实际应用程序。

```
[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()
```

此代码是一个非常小的Python Web服务器，您在webserver.py中创建它。Web服务器只需在请求GET时立即返回IOx python webserver。Web服务器启动的端口可以是端口80，也可以是指定给webserver.py的第一个参数。

在运行函数中，此代码还包含对日志文件的写入。日志文件可从IOx客户端或本地管理器中查看。

4.创建Dockerfile和Docker容器。

既然您的容器中应该运行应用程序(webserver.py)，现在应该构建Docker容器了。容器在Dockerfile中定义：

```
[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
```

如您所见，Dockerfile也保持简单。从Alpine基础映像开始，安装Python并将您的webserver.py复制到容器的根目录。

准备好您的Dockerfile后，您可以构建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命令按照您在Dockerfile中的请求下载基本映像并安装Python和依赖关系。最后一个命令用于验证。

5.测试创建的Docker容器。

此步骤是可选的，但最好验证刚刚构建的Docker容器是否准备按预期工作。

```
[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的输出中可以看到，启动webserver.py后，它会侦听端口9000。

6.使用Docker容器创建IOx包。

现在，您已验证容器中Web服务器的功能，现在应准备并构建IOx包进行部署。由于Dockerfile提供了构建Docker容器的说明，package.yaml提供了IOx客户端构建IOx软件包的说明。

```
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"]
```

有关package.yaml内容的详细信息，请访问：https://developer.cisco.com/media/iox-dev-guide-3-10-16/concepts/package_descriptor/。

创建package.yaml后，您可以开始生成IOx程序包。

第一步是导出Docker映像的根FS:

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

接下来，您可以生成软件包.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 definit
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
```

生成结果是一个IOx包(package.tar)，它包含Docker容器，准备部署在IOx上。

注意：IOxclient也可以一步完成docker save命令。在CentOS上，这会导致导出到默认的rootfs.img而不是rootfs.tar，后者会在后面的过程中产生问题。只需使用IOx客户端docker软件包IOxpythonweb:1.0即可完成创建这一步。

8.在IOx设备上部署、激活和启动软件包。

最后一步是将IOx包部署到IOx设备，激活后启动。这些步骤可以使用IOx客户端、本地管理器或雾网络导向器来完成。对于本文而言，您可以使用IOx客户端。

要将软件包部署到IOx设备，请使用名称python_web:

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

在激活应用之前，您需要定义网络配置方式。为此，您需要创建一个JSON文件。激活后，可以将其附加到激活请求。

```
[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"},
    }
}
[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
```

此处的最后一项操作是启动您刚刚部署和激活的应用程序：

```
[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
```

由于已将IOx应用配置为在端口9000上侦听端口9000上的tor HTTP请求，因此仍需要将该端口从

IOx设备转发到容器，因为容器位于NAT之后。在Cisco IOS®上执行此操作。

```
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
```

第一个命令列出了GOS的内部IP地址（负责启动/停止/运行IOx容器）。

第二个命令为IOS端的Gi0接口上的端口9000配置一个到GOS的静态端口转发。如果您的设备通过L2端口连接（很可能是IR829上的情况），则需要用配置了ip nat outside语句的正确VLAN替换Gi0接口。

验证

使用本部分可确认配置能否正常运行。

为了验证Web服务器是否正常运行和响应，您可以尝试使用此命令访问Web服务器。

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

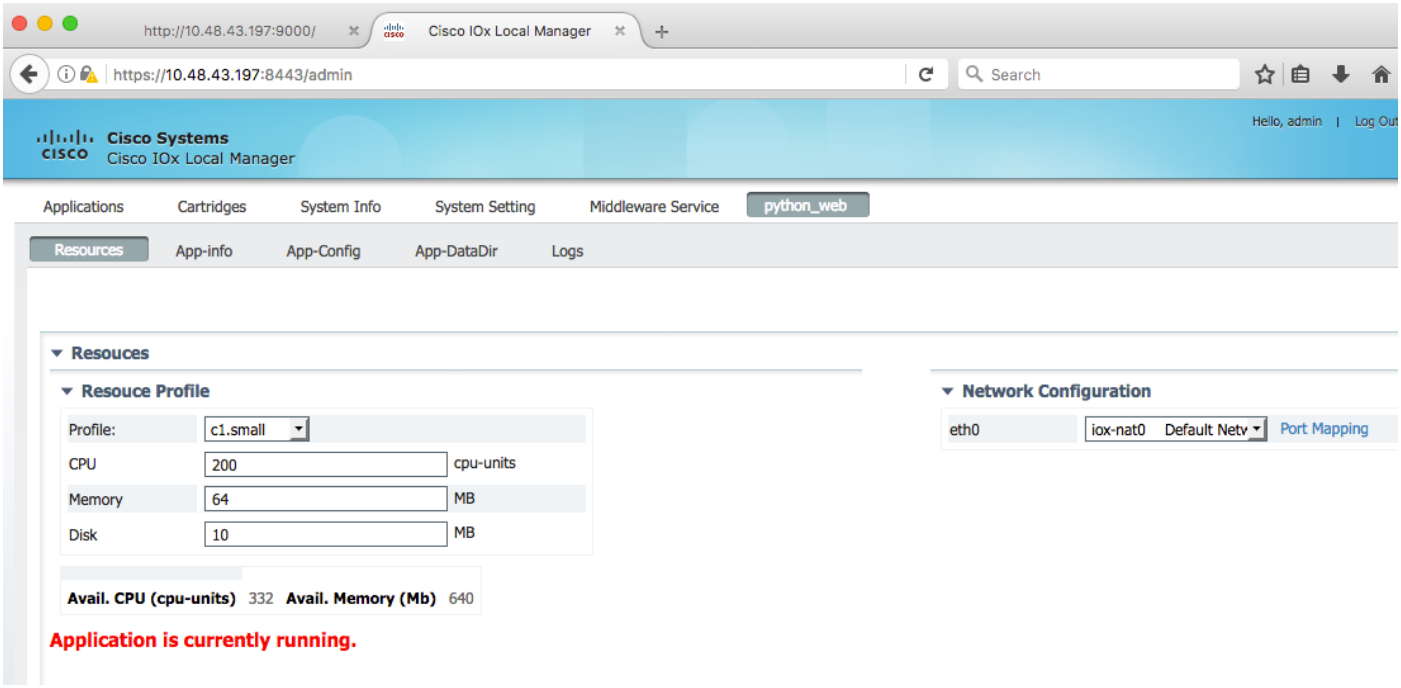
或者，从如图所示的实际浏览器中。



您还可以从IOxclient CLI验证应用状态：

```
[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
```

您还可以从本地管理器GUI验证应用状态，如图所示。



要查看您在webserver.py中写入的日志文件：

```
[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
```

故障排除

本部分提供了可用于对配置进行故障排除的信息。

要对应用和/或容器进行故障排除，最简单的方法是连接到运行以下内容的应用的控制台：

```
[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
```

关于此翻译
思科采用人工翻译与机器翻译相结合的方式将此文档翻译成不同语言，希望全球的用户都能通过各自的语言得到支持性的内容。

请注意：即使是最好的机器翻译，其准确度也不及专业翻译人员的水平。

Cisco Systems, Inc. 对于翻译的准确性不承担任何责任，并建议您总是参考英文原始文档（已提供链接）。