配置在IOx的一小高山Linux码头工人镜像

目录

简介

本文描述配置过程创建，实施和管理在思科IOX有能力设备的基于码头工人的应用程序。

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IOx有能力设备：
  配置的IP地址访客操作系统(GOS)和Cisco应用框架(CAF)运行为对CAF (端口8443)的访问配置的网络地址转换(NAT)为对GOS shell (端口2222)的访问配置的NAT
- Linux主机(最小CentOS 7安装使用此条款)
- IOx能下载的客户端安装文件：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能主机不同种类的包Java、Python、主要LXC，虚拟机等，并且能也运行码头工人容器。思科提供基本镜像和一个完整码头工人集线器信息库： 能使用构件码头工人容器的https://devhub.cisco.com/artifactory/webapp/#/artifacts/browse/tree/General/iox-docker。

这是关于怎样的一分步指南构件有使用的一个简单码头工人容器高山Linux。高山Linux是一小Linux镜像(在5MB附近)，是常用的作为码头工人容器的一个基础。在此条款，您从一个已配置的

IOx设备开始，空CentOS 7 Linux计算机和您在码头工人容器在IOx设备建立一小Python Web服务器，包它并且部署那。

配置

1. 安装并且准备Linux主机的IOx客户端。

IOx客户端是能包应用程序和与IOX有能力设备联络管理IOx应用程序的工具。

在您下载ioxclient安装包后，可以安装如下：

```
[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
```

正如你看到的在IOx客户端第一启动，配置文件可以为您能管理与IOx客户端的IOx设备生成。万一希望执行此以后或，如果想要添加/更改设置，您能运行后此的命令：**ioxclient配置文件创建**

2. 安装并且准备Linux主机的码头工人。

码头工人用于构件容器和测试我们的示例应用的执行。

安装码头工人的安装步骤非常取决于您安装它的Linux OS。对于此条款，您能使用CentOS 7。对于不同的分配的安装说明，参考： https://docs.docker.com/engine/installation/。

安装前提条件：

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

添加码头工人回购：

```
[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
```

安装码头工人(请接受GPG密钥验证，当您安装)：

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

启动码头工人：

```
[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
```

为了能访问/运行码头工人作为一个普通用户，请添加此用户到码头工人组并且刷新组成员：

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

登陆到码头工人集线器：

码头工人集线器包含您能使用的高山基本镜像。万一没有码头工人ID，您需要注册：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, head over to https://hub.docker.com to create one.
Username: jensdepuydt
Password:
Login Succeeded
```

3. 创建Python Web服务器。

即然准备完成，您能开始建立在IOX enable (event)设备能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()
```

此代码是一非常最小Python Web服务器，您在webserver.py创建。当GET是请求的， Web服务器返回IOx Python网络服务器。Web服务器启动的端口可以是端口80或第一个参数给对webserver.py。

此代码也在运行功能包含，写入到日志文件。日志文件为从IOx客户端或当地干事的咨询是可用的。

4. 创建Dockerfile和码头工人容器。

即然您有在您的容器应该运行的应用程序(webserver.py)，是时间构件码头工人容器。容器在Dockerfile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
```

正如你看到的Dockerfile也被保持简单化。您从高山基本镜像开始，安装Python并且复制您的webserver.py到容器的根。

一旦有就绪您的Dockerfile，您能构件码头工人容器：

```
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
```

码头工人构建命令下载基本镜像并且安装Python，并且从属关系，作为您在Dockerfile请求。最后命令是为验证。

5. 测试已创建码头工人容器。

此步骤可选，但是验证是好的您的被构件的码头工人容器准备工作正如所料。

```
[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
```

正如你在netstat输出看到，在您开始webserver.py后，在端口9000侦听。

6. 创建IOx包用码头工人容器。

即然您验证您的在容器的Web服务器的功能，是时间准备和建立部署的IOx包。当Dockerfile提供说明构件码头工人容器， package.yaml为IOx客户端提供说明建立您的IOx包。

```
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"]
```

可以找到关于package.yaml的内容的更多信息此处： https://developer.cisco.com/media/iox-dev-

guide-3-10-16/concepts/package_descriptor/。

在您创建package.yaml后，您能开始建立IOx包。

第一步将导出码头工人镜像的根FS ：

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

其次，您能构件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
```

构建的结果是IOx包(package.tar)，包含码头工人容器，就绪部署在IOx。

> **注意**：IOxclient能执行码头工人save命令在一个步骤。在CentOS，这发生导出到默认rootfs.img而不是rootfs.tar，给麻烦后在进程。创建的这一个步骤可以实行与使用： IOx客户端码头工人包IOxpythonweb:1.0。

8. 部署，激活并且启动在IOx设备的包。

最后步骤将部署IOx包对IOx设备，激活它和开始。这些步骤可能实行与使用IOx客户端、当地干事或者雾网络导控器。对于此条款，您能使用IOx客户端。

为了部署包到IOx设备，请使用命名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
```
在您能激活应用程序前，您需要定义网络配置如何是。要执行如此，您需要创建JSON文件。当您激活时，它可以附加到激活请求。

```
[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"}]
    }
}
[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
```
此处上一操作是运行您实施并且激活的应用程序：

```
[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
```
因为您在突岩HTTP请求的端口9000配置您的IOx应用程序侦听，您仍然需要转发从您的IOX设备的端口到容器作为容器是在NAT后。执行此在Cisco IOS如此执行。

```
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
```
第一条命令列出GOS内部IP地址(负责对开始/停/请运行IOx容器)。

第二条命令配置端口的9000一静态端口转发IOS旁拉的Gi0接口的对GOS。万一您的设备通过连接是很可能在IR829的案件)的L2端口(您需要由有配置的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。

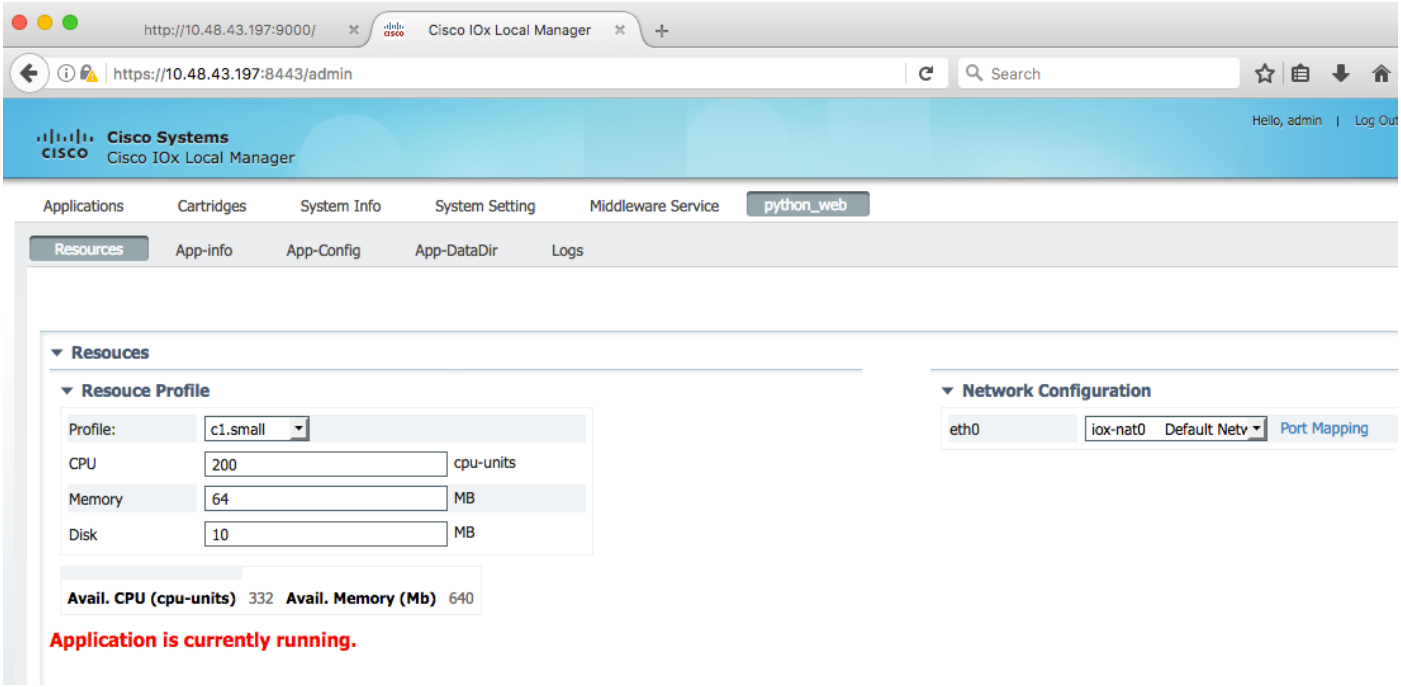
为了验证，如果Web服务器适当地运行并且响应，您能设法访问Web服务器用此命令。

```
[jedepuyd@db iox_docker_pythonweb]$ curl http://10.48.43.197:9000/
<html><body><h1>IOX python webserver</h1></body></html>
```
或者，从如镜像所显示的一个实时浏览器。

您能也验证从IOxclient CLI的应用程序状态：

```
[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
```

如镜像所显示，并且您能也验证从本地管理器GUI的应用程序状态。



为了查看一下把写您到在webserver.py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。

为了排除故障应用程序和容器，简便的方法是连接对运行应用程序的控制台：

```
[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
```