Настройте маленький альпийский образ докера Linux на IOx

Содержание

Введение

Этот документ описывает процесс конфигурирования, чтобы создать, развернуть и управлять Основанными на докере приложениями на IOx-устройствах-с-поддержкой Cisco.

Предварительные условия

Требования

Для этого документа отсутствуют особые требования.

Используемые компоненты

Сведения, содержащиеся в данном документе, касаются следующих версий программного обеспечения и оборудования:

- Устройство с поддержкой IOx, которое настроено для IOx:
  IP-адрес настроенГостевая операционная система (GOS) и выполнение Платформы приложения Cisco (CAF)Технология NAT настроила для доступа к CAF (порт 8443)NAT настроил для доступа к оболочке GOS (порт 2222)
- Хост Linux (минимальная установка CentOS 7 используется для этой статьи),
- Файлы установки клиентской части IOx, которые могут быть загружены от: [https://программное обеспечение. cisco . com/download/release.html?mdfid=286306005&softwareid=286306762](#)

Сведения, представленные в этом документе, были получены от устройств, работающих в специальной лабораторной среде. Все устройства, описанные в этом документе, были запущены с чистой (стандартной) конфигурацией. В рабочей сети необходимо изучить потенциальное воздействие всех команд до их использования.

Общие сведения

IOx может разместить различные типы пакетов в основном Java, Python, LXC, Виртуальная машина (VM) и т.д., и это может также выполнить контейнеры Докера. Cisco предлагает базовый образ и завершенный репозиторий концентратора Докера: https://devhub. cisco . com/artifactory/webapp/#/artifacts/browse/tree/General/iox-docker, который может использоваться для построения контейнеров Докера.

Вот пошаговые инструкции о том, как создать простой контейнер Докера с использованием альпийского Linux. Альпийский Linux является маленьким образом Linux (приблизительно 5 МБ), который часто используется в качестве ядра для контейнеров Докера. В этой статье вы запускаете с настроенного устройства IOx, пустой машины CentOS 7 Linux, и вы создаете маленький Web-сервер Python, упаковываете его в контейнере Докера и развертываете это на устройстве IOx.

Настройка

1. Установите и подготовьте клиента IOx на хосте Linux.

Клиент IOx является программным средством, которое может упаковать приложения и связаться с IOx-устройством-с-поддержкой для управления приложениями IOx.

После загрузки ioxclient пакета установки он может быть установлен следующим образом:

```
[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
```

Как вы можете видеть, на первом запуске клиента IOx, профиль может генерироваться для устройства IOx, которым можно управлять с клиентом IOx. В случае, если требуется сделать это позже или если вы хотите добавить/изменить параметры настройки, можно выполнить эту команду позже: **профили ioxclient создают**

2. Установите и подготовьте Докера на хосте Linux.

Докер используется, чтобы создать контейнер и протестировать выполнение нашего

примера приложения.

Шаги установки для установки Докера в большой степени зависят от Linux OS, на котором вы устанавливаете его. Для этой статьи можно использовать CentOS 7. Для инструкций по установке для других дистрибутивов обратитесь к: https://docs.docker.com/engine/installation/.

Предварительные условия установки:

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

Добавьте Докера repo:

```
[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
```

Установите Докера (примите ключевую проверку GPG, когда вы устанавливаете):

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

Запустите докера:

```
[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
```

Чтобы быть в состоянии обратиться/выполнить к Докеру как к стандартному пользователю, добавьте этого пользователя к группе Докера и обновите состав группы:

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

Войдите к концентратору докера:

Концентратор докера содержит альпийский базовый образ, который можно использовать. В случае, если у вас еще нет ID Докера, необходимо зарегистрироваться на: 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. Создайте Web-сервер Python.

Теперь, когда подготовка сделана, можно начать создавать реальное приложение, которое может работать на IOx-разрешать устройстве.

```
[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()
```

Этот код является очень минимальным Web-сервером Python, который вы создаете в webserver.py. Web-сервер просто возвращает веб-сервер питона IOx, как только запрашивают GET. Порт, на котором запуски Web-сервера могут или быть портом 80 или первым аргументом, данным webserver.py.

Этот код также содержит, в функции выполнения, записи к файлу журнала. Файл журнала доступен для консультации от клиентского или локального менеджера IOx.

4. Создайте контейнер Dockerfile и Docker.

Теперь, когда у вас есть приложение (webserver.py), который должен работать в вашем контейнере, пора создать контейнер Докера. Контейнер определен в 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
```

Как вы можете видеть Dockerfile также сохранен простым. Вы запускаете с альпийского базового образа, устанавливаете Python и копируете свой webserver.py к root контейнера.

Как только у вас есть свой готовый Dockerfile, можно создать контейнер Докера:

```
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
```

Команда сборки Докера загружает базовый образ и устанавливает Python и зависимости, когда вы запросили в Dockerfile. Последняя команда для проверки.

5. Протестируйте созданный контейнер Докера.

Этот шаг является дополнительным, но хорошо проверить, что ваш просто созданный контейнер Докера готов работать как ожидалось.

```
[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
```

Как вы можете видеть в выходных данных netstat после начала webserver.py он слушает на порту 9000.

6. Создайте пакет IOx с контейнером Докера.

Теперь, когда вы проверили функциональность своего Web-сервера в контейнере, пора подготовить и создать пакет IOx для развертываний. Поскольку Dockerfile предоставляет инструкции для построения контейнера Докера, package.yaml предоставляет инструкции для

клиента IOx для построения пакета 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"]
```

Дополнительные сведения о содержании package.yaml могут быть найдены здесь: https://разработчик. cisco . com/media/iox-dev-guide-3-10-16/concepts/package_descriptor/.

После создания package.yaml можно начать создавать пакет IOx.

Первый шаг должен экспортировать корневой FS образа Докера:

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

Затем, можно создать 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
```

Результатом сборки является пакет IOx (package.tar), который содержит контейнер Докера, готовый быть развернутым на IOx.

> **Примечание:** IOxclient может сделать команду save докера за один шаг также. На CentOS это заканчивается для экспортирования в по умолчанию rootfs.img вместо rootfs.tar, который дает проблему позже в процессе. Один шаг для создания может быть выполнен с использованием: пакет докера клиента IOx IOxpythonweb:1.0.

8. Разверните, активируйте и запустите пакет на устройстве IOx.

Последние шаги должны развернуть пакет IOx на устройстве IOx, активировать его и запуститься. Эти шаги могут или быть выполнены с использованием клиента IOx, Локального Управляющего узла Сети Менеджера или Вуали. Для этой статьи можно использовать клиента IOx.

Для развертывания пакета на устройстве IOx используйте название 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
```

Прежде чем можно будет активировать приложение, необходимо определить, как конфигурация сети была бы. Для этого необходимо создать файл JSON. Когда вы активируете, это может быть присоединено к запросу активации.

```
[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
```

Последнее действие здесь должно запустить приложение, которое вы просто развернули и активировали:

```
[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
```

Так как вы настроили свое приложение IOx для прослушивания на порту 9000 для запросов HTTP скалистой вершины, все еще необходимо передать тот порт от IOx-устройства до контейнера, как контейнер находится позади NAT. Выполните это на Cisco IOS®, чтобы сделать так.

```
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
```

Первые списки команд внутренний IP-адрес GOS (ответственный за запускают/останавливают/выполняют контейнеры IOx).

Вторая команда настраивает статический порт вперед для порта 9000 на интерфейсе Gi0 стороны IOS к GOS. В случае, если ваше устройство связано через порт L2 (который, скорее всего, имеет место на IR829), необходимо заменить интерфейс Gi0 корректной VLAN, которой настроили оператор ip nat outside.

Проверка

Воспользуйтесь данным разделом для проверки правильности функционирования вашей конфигурации.

Чтобы проверить, выполняется ли Web-сервер и отвечает должным образом, можно попытаться обратиться к Web-серверу с этой командой.

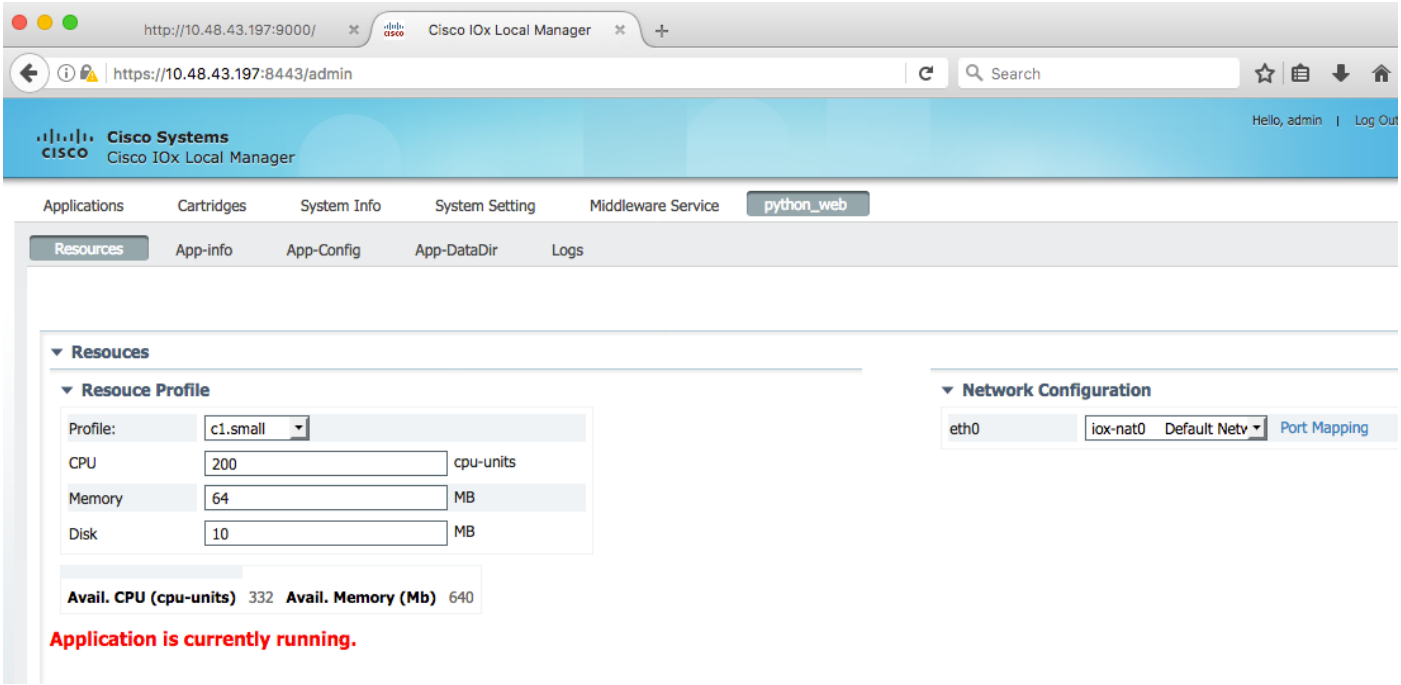
```
[jedepuyd@db iox_docker_pythonweb]$ curl http://10.48.43.197:9000/
<html><body><h1>IOX python webserver</h1></body></html>
```

Или, от реального браузера как показано в образе.

Можно также проверить состояние приложения от CLI 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
```

и можно также проверить состояние приложения от Локального Графического интерфейса пользователя менеджера как показано в образе.



Чтобы взглянуть на файл журнала, в который вы пишете в 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
```

Устранение неполадок

Этот раздел обеспечивает информацию, которую вы можете использовать для того, чтобы устранить неисправность в вашей конфигурации.

Для устранения проблем приложения и/или контейнера, самый легкий путь состоит в том, чтобы соединиться с консолью приложения, которое выполняется:

```
[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
```