Configurar uma imagem alpina pequena do estivador de Linux em IOx

Índice

Introdução

Este documento descreve o processo de configuração para criar, distribuir e controlar aplicativos Estivador-baseados em dispositivos IOx-capazes de Cisco.

Pré-requisitos

Requisitos

Não existem requisitos específicos para este documento.

Componentes Utilizados

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

- Dispositivo capaz de IOx que é configurado para IOx:
  Endereço IP de Um ou Mais Servidores Cisco ICM NT configuradoSistema operacional do convidado (GOS) e de estrutura do aplicativo Cisco ser executado (CAF)Network Address Translation (NAT) configurado para o acesso a CAF (porta 8443)NAT configurado para o acesso ao shell GOS (porta 2222)
- Host de Linux (uma instalação mínima de CentOS 7 é usado para este artigo)
- Arquivos da instalação de cliente de IOx de que pode ser transferido: https://software.cisco.com/download/release.html?mdfid=286306005&softwareid=286306762

As informações neste documento foram criadas a partir de dispositivos em um ambiente de laboratório específico. Todos os dispositivos utilizados neste documento foram iniciados com uma configuração (padrão) inicial. Se a sua rede estiver ativa, certifique-se de que entende o impacto potencial de qualquer comando.

Informações de Apoio

IOx pode hospedar tipos diferentes das Javas dos pacotes principalmente, do pitão, do LXC, da máquina virtual (VM) etc., e pode igualmente executar recipientes do estivador. Cisco oferece uma imagem de base e um repositório completo do hub do estivador: https://devhub.cisco.com/artifactory/webapp/#/artifacts/browse/tree/General/iox-docker que pode ser usado para construir recipientes do estivador.

Está aqui um guia passo a passo em como construir um recipiente simples do estivador com o uso de Linux alpino. Linux alpino é uma imagem pequena de Linux (em torno de 5MB), que seja usada frequentemente como uma base para recipientes do estivador. Neste artigo, você parte de um dispositivo configurado de IOx, uma máquina e você vazios de CentOS 7 Linux constroem um servidor de Web pequeno do pitão, empacotam-no em um recipiente do estivador e distribuem-nos aquele em um dispositivo de IOx.

Configurar

1. Instale e prepare o cliente de IOx no host de Linux.

O cliente de IOx é a ferramenta que pode empacotar aplicativos e se comunicar com o dispositivo IOx-capaz para controlar aplicativos de IOx.

Depois que você transfere o pacote ioxclient da instalação, pode ser instalado como segue:

```
[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, no primeiro lançamento do cliente de IOx, um perfil pode ser gerado para o dispositivo de IOx que você pode controlar com cliente de IOx. Caso que você gostaria de fazer este mais atrasado ou se você quer adicionar/mudança os ajustes, você pode executar este comando mais tarde: **os perfis ioxclient criam**

2. Instale e prepare o estivador no host de Linux.

O estivador é usado para construir um recipiente e para testar a execução de nosso aplicativo de

amostra.

As etapas de instalação para instalar o estivador dependem pesadamente do Linux OS que você o instala sobre. Para este artigo, você pode usar CentOS 7. Para instruções de instalação para distribuições diferentes, refira: https://docs.docker.com/engine/installation/.

Instale condições prévias:

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

Adicionar o repo do estivador:

```
[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 estivador (aceite a verificação da chave GPG quando você instala):

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

Estivador do começo:

```
[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
```

A fim poder alcançar/estivador executado como um usuário regular, adicionar este usuário ao grupo do estivador e refresque a membrasia do clube:

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

Entre ao hub do estivador:

O hub do estivador contém a imagem de base alpina que você pode usar. Caso que você não tem um estivador ID ainda, você precisa de registrar-se sobre: 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 de Web do pitão.

Agora que a preparação é feita, você pode começar construir o aplicativo real que pode ser executado no dispositivo da IOx-possibilidade.

```
[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 de Web muito mínimo do pitão, que você crie em webserver.py. O servidor de Web retorna simplesmente o web server do pitão de IOx assim que um GET for pedido. A porta em que o servidor de Web começa pode ser a porta 80 ou o primeiro argumento dado a webserver.py.

Este código igualmente contém, na função da corrida, uma escrita a um arquivo de registro. O arquivo de registro está disponível para a consulta do cliente de IOx ou do gerente local.

4. Crie o recipiente de Dockerfile e de estivador.

Agora que você tem o aplicativo (webserver.py) que deve ser executado em seu recipiente, é hora de construir o recipiente do estivador. Um recipiente é definido em um 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 Dockerfile é mantido igualmente simples. Você começa com a imagem de base alpina, instala o pitão e copia seu webserver.py à raiz do recipiente.

Uma vez que você tem seu Dockerfile pronto, você pode construir o recipiente do estivador:

```
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 da construção do estivador transfere a imagem de base e instala o pitão e as dependências, como você pediram no Dockerfile. O último comando é para a verificação.

5. Teste o recipiente criado do estivador.

Esta etapa é opcional mas é bom verificar que seu recipiente construído justo do estivador está pronto para trabalhar como 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 começa o webserver.py, escuta na porta 9000.

6. Crie o pacote de IOx com o recipiente do estivador.

Agora que você verificou a funcionalidade de seu servidor de Web no recipiente, é hora de preparar e construir o pacote de IOx para o desenvolvimento. Enquanto o Dockerfile fornece instruções para construir um recipiente do estivador, o package.yaml fornece instruções para que o cliente de IOx construa seu pacote de 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 informação nos índices do package.yaml pode ser encontrada aqui: https://developer.cisco.com/media/iox-dev-guide-3-10-16/concepts/package_descriptor/.

Depois que você cria o package.yaml, você pode começar construir o pacote de IOx.

A primeira etapa é exportar a raiz FS da imagem do estivador:

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

Em seguida, você pode construir 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
```

O resultado da construção é um pacote de IOx (package.tar), que contenha o recipiente do estivador, pronto para ser distribuído em IOx.

> Nota: IOxclient pode fazer o comando save do estivador em uma etapa também. Em CentOS, isto resulta para exportar para o padrão rootfs.img em vez de rootfs.tar, que dá o problema mais tarde no processo. A uma etapa a criar pode ser executada com o uso de: Pacote IOxpythonweb:1.0 do estivador do cliente de IOx.

8. Distribua, ative e comece o pacote no dispositivo de IOx.

As últimas etapas são distribuir o pacote de IOx ao dispositivo de IOx, ativá-lo e começá-lo. Estas etapas podem ser executadas com o uso do cliente de IOx, do gerente local ou do diretor da rede da névoa. Para este artigo, você pode usar o cliente de IOx.

A fim distribuir o pacote ao dispositivo de IOx, use o python_web do nome:

```
[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 que você possa ativar o aplicativo, você precisa de definir como a configuração de rede seria. Para fazer assim, você precisa de criar um arquivo JSON. Quando você ativa, pode ser anexado ao pedido da 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"}]
    }
}
[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 começar o aplicativo que você apenas distribuiu e ativou:

```
[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
```

Desde que você configurou seu pedido de IOx escutar na porta 9000 para pedidos do HTTP do tor, você ainda precisa de enviar que a porta de seu IOx-dispositivo ao recipiente como o recipiente é atrás do NAT. Execute isto em Cisco IOS® para fazer assim.

```
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
```

As primeiras lista de comando o endereço IP interno do GOS (o responsável para o começo/parada/executa os recipientes de IOx).

O comando second configura uma porta estática para a frente para a porta 9000 na relação Gi0 do IO-lado ao GOS. Caso que seu dispositivo está conectado através de uma porta L2 (que é mais provável o caso em IR829), você precisa de substituir a relação Gi0 pelo VLAN correto que tem a indicação exterior nat IP configurada.

Verificar

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

A fim verificar se o servidor de Web é executado e responde corretamente, você pode tentar alcançar o servidor de Web com este comando.

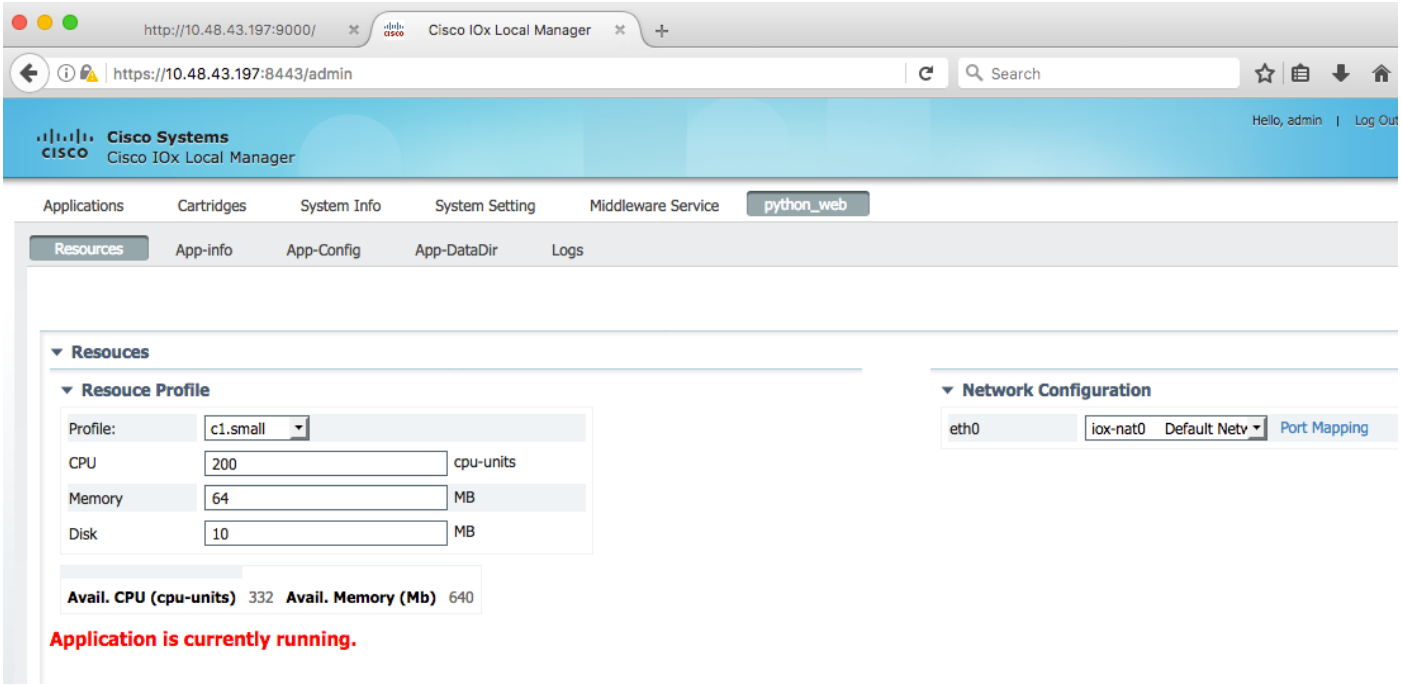
```
[jedepuyd@db iox_docker_pythonweb]$ curl http://10.48.43.197:9000/
<html><body><h1>IOX python webserver</h1></body></html>
```

Ou, de um navegador real segundo as indicações da imagem.

Você pode igualmente verificar o estado do aplicativo do IOxclient CLI:

```
[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 pode igualmente verificar o estado do aplicativo do GUI de gerenciador local segundo as indicações da imagem.



A fim ter um olhar no arquivo de registro que você redige em 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
```

Troubleshooting

Esta seção fornece a informação que você pode se usar a fim pesquisar defeitos sua configuração.

A fim pesquisar defeitos o aplicativo e/ou o recipiente, a maneira a mais fácil é conectar ao console do aplicativo que é executado:

```
[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
```