Een IOx-toepassing implementeren met IOxClient

Inhoud

Inleiding

Dit document beschrijft de implementatie van een toepassing met behulp van ioxclient.

Doel

Dit document is bedoeld om inzicht te krijgen in de implementatie van een toepassing met behulp van ioxclient.

De focus van dit document is volledig praktisch, dus als u meer technische details wilt, raad ik u aan de gedeelde documentatie te bekijken.

Voorwaarden

- Basiskennis van de Cisco IOS XE en Cisco IOS besturingssystemen.
- Een goed begrip van de levenscyclus van Docker en containers.
- Basis Linux-bewerkingen.

Vereisten

Bevestig dat als uw apparaat iox ondersteunt, u de compatibiliteitsmatrix kunt controleren: Platform Support Matrix

Download ook iox-client volgens uw pc-specificaties: Downloads

Gebruikte componenten

De informatie in dit document is gebaseerd op deze software- en hardwareversies:

- ioxclient versie 1.17.0.0
- Router C8000v, versie 17.12.3a
- Ubuntu Machine versie 10 .04
- Installeer Docker Engine versie 24.0.9 of ouder.

De informatie in dit document is gebaseerd op de apparaten in een specifieke laboratoriumomgeving. Alle apparaten die in dit document worden beschreven, hadden een opgeschoonde (standaard)configuratie. Als uw netwerk live is, moet u zorgen dat u de potentiële impact van elke opdracht begrijpt.

Overzicht

Wat is IOx?

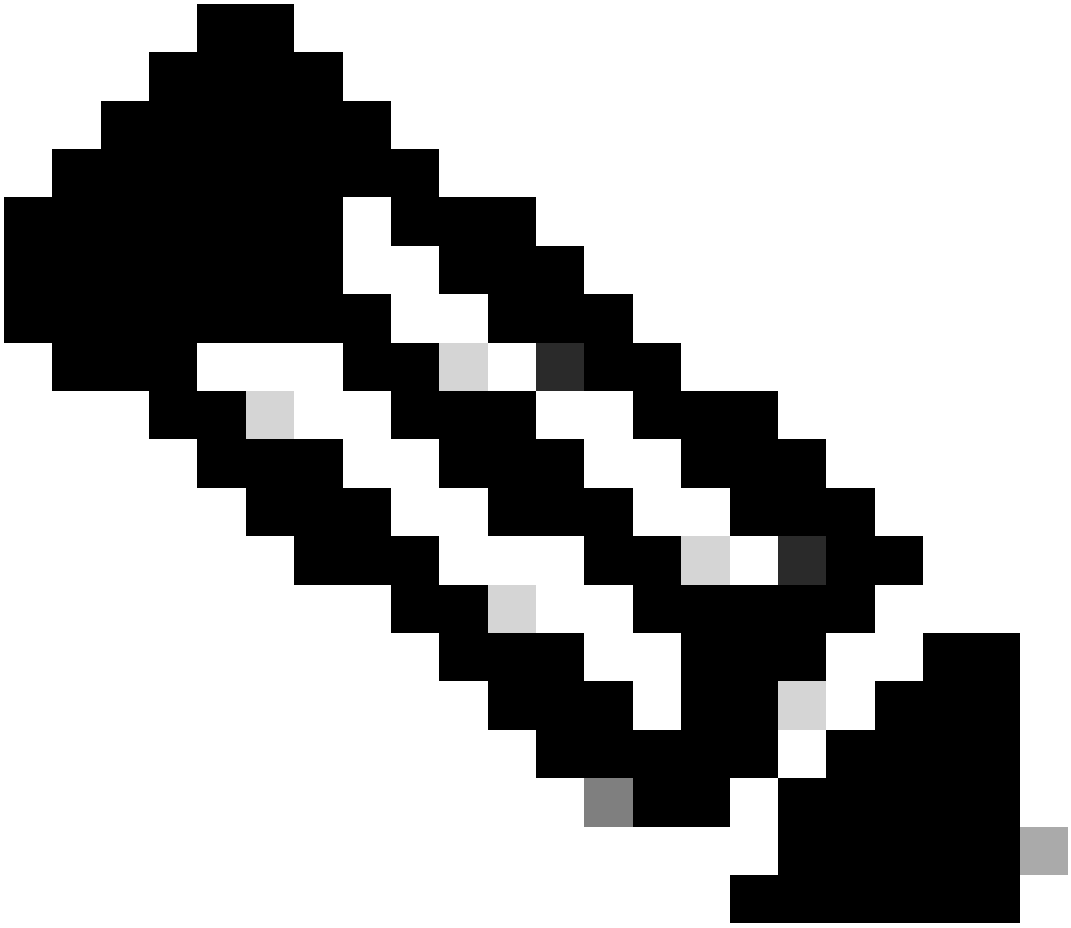
IOx is de toepassingsomgeving voor Cisco-apparaten, met deze functie kunnen we de toepassingen verpakken in een formaat dat compatibel is met IOx, met behulp van tools zoals ioxclient.

Wat is een ioxclient?

ioxclient is een opdrachtregelprogramma dat deel uitmaakt van de Cisco IOx SDK. Het wordt gebruikt voor het ontwikkelen, testen en implementeren van IOx-toepassingen op Cisco IOx-apparaten.

Definieer de applicatie

Deze voorbeeldcode maakt een basis-HTTP-server die luistert via poort 8000. Deze code dient als de kernfunctionaliteit van het Docker-image (Dockerfile).



Opmerking: een Dockerfile is alleen vereist bij het ontwikkelen van aangepaste images met specifieke functionaliteit. Een applicatie kan worden getrokken uit een container repository, geëxporteerd en gebruikt als basis voor ioxclient.

Python-toepassing

```
import http.server
import socketserver

PORT = 8000
Handler = http.server.SimpleHTTPRequestHandler

with socketserver.TCPServer(("", PORT), Handler) as httpd:
    print(f"Serving at port {PORT}")
    httpd.serve_forever()
```

Het Docker-bestand definiëren

```
FROM python:alpine3.20
WORKDIR /apps
COPY . .
EXPOSE 8000
ENTRYPOINT [ "python" ]
CMD [ "main.py" ]
```

Met deze code stelt u een http-server in die luistert via poort 8000 en verpakt u de toepassing in een Docker-bestand.

pakket

Het is vereist om een package.YAML-bestand met metagegevens en brondefinities te maken en in te vullen voor een juiste implementatie van de toepassing.

Het YAML-bestand is een formaat, dit formaat is aantrekkelijk vanwege de eenvoudige syntaxis, in het bestand kunnen we aspecten van de toepassing specificeren als omgevingsvariabelen, poorten, afhankelijkheden enzovoort.

Voorbeeld van YAML uit de Cisco DevNet IOx Template Repository

```
descriptor-schema-version: "2.2"

info:
  name: iox_docker_python
  description: "IOx Docker Python Sample Application"
  version: "1.0"
  author-link: "http://www.cisco.com"
  author-name: "Cisco Systems"

app:
  cpuarch: "x86_64"
  type: docker
  resources:
    profile: c1.small

  # Specify runtime and startup
  startup:
    rootfs: rootfs.tar
    target: ["python3 main.py"]
```

Raadpleeg de documentatie om de geldige waarden in het pakketbestand te raadplegen:

- Apparaatdocumentatie: IOx-pakketbeschrijving
- GitHub repository: CiscoDevNet / iox-app-template

Voor dit document bevat het configuratiebestand van YAML de volgende informatie:

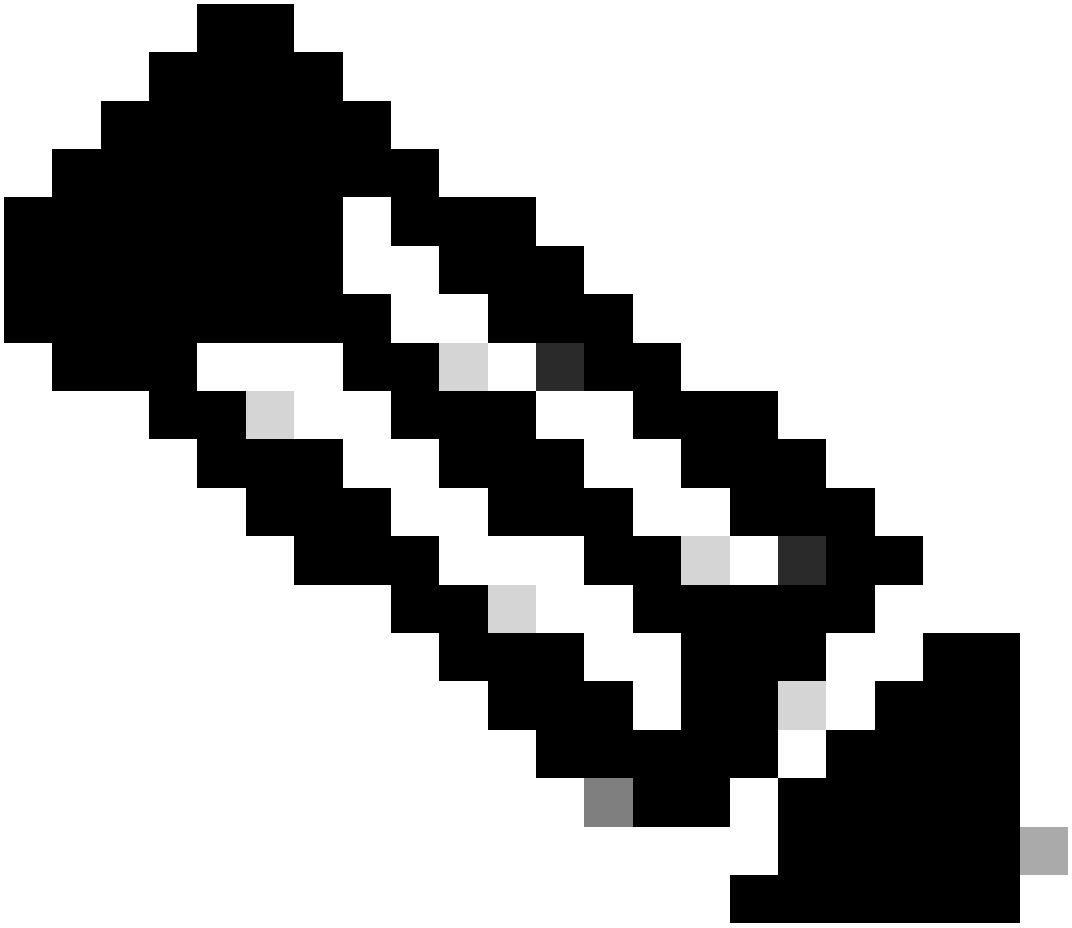
```
descriptor-schema-version: "2.2"

info:
  name: "tac_app"
  description: "tac_app"
  version: "1.0"
  author-name: "TAC-TEST"

app:
  cpuarch: x86_64
  type: docker
  resources:
    profile: "custom"
    cpu: 100                # CPU en MHz assigned to the application.
    disk: 50               # Storage in MB for the disk
    memory: 128           # Memory en MB assigned to the application.
    network:
      -
        interface-name: eth0
        ports:
          tcp:
            - 8000
  startup:
    rootfs: "rootfs.tar"         # Container file system
    target: "python main.py"     # Command to start the application
```

Vanwege een incompatibiliteit tussen Docker Engine versie 25.0 en ioxclient, is de aanbevolen aanpak om een Linux-distributie te gebruiken die Docker Engine versie 24.0.9 of eerder ondersteunt, omdat versie 24.0.9 de nieuwste ondersteunde versie is voor compatibiliteit met ioxclient.

In dit voorbeeld is het Docker-image dat wordt gebruikt om de IOx-clientfunctionaliteit aan te tonen, gebouwd op een op Ubuntu gebaseerde virtuele machine waarop versie 10 .04 wordt uitgevoerd. Dit image is specifiek gekozen omdat de binaire bestanden van Docker Engine .deb beschikbaar zijn voor deze distributie/versie.



Opmerking: De enige manier om oudere versies van Docker te installeren, is door het via de bin-bestanden te installeren. Deze binaries zijn specifieke versies van de software die al zijn voorbereid om rechtstreeks op een bepaald besturingssysteem te draaien.

Configuratie

Om de VM voor te bereiden met de genoemde specificaties, gaat u verder met het installeren van het binaire bestanden van een oude versie van Docker:

```
wget https://download.docker.com/linux/ubuntu/dists/focal/pool/stable/amd64/docker-ce-cli_24.0.9-1~ubun
wget https://download.docker.com/linux/ubuntu/dists/focal/pool/stable/amd64/docker-ce_24.0.9-1~ubuntu.2
wget https://download.docker.com/linux/ubuntu/dists/focal/pool/stable/amd64/docker-buildx-plugin_0.11.2
wget https://download.docker.com/linux/ubuntu/dists/focal/pool/stable/amd64/docker-compose-plugin_2.21.
wget https://download.docker.com/linux/ubuntu/dists/focal/pool/stable/amd64/containerd.io_1.7.19-1_amd6

And installed them:
sudo dpkg -i ./containerd.io_1.7.19-1_amd64.deb \
```

```
./docker-ce_24.0.9-1~ubuntu.20.04~focal_amd64.deb \
./docker-ce-cli_24.0.9-1~ubuntu.20.04~focal_amd64.deb \
./docker-buildx-plugin_0.11.2-1~ubuntu.20.04~focal_amd64.deb \
./docker-compose-plugin_2.21.0-1~ubuntu.20.04~focal_amd64.deb
```

Zodra alle bestanden zijn geïnstalleerd, is de machine klaar om de iox-toepassing te verpakken.

verpakkingsprocédé

Breng de Python-code en het Docker-bestand over naar de virtuele machine, controleer of beide bestanden zich in dezelfde map bevinden en ga vervolgens verder met het maken van het Docker-image:

```
sudo docker build -t tac_app .
```

Voer de onderstaande opdracht uit om de Docker-images weer te geven die beschikbaar zijn in de lokale systeemopslagplaats:

```
ubuntu@ip-172-31-30-249:~$ sudo docker images

REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
tac_app       latest    94a1c2ba4b08   19 seconds ago   1.78GB
```

Vanaf hier zijn er 2 alternatieven

1 - Pak de toepassing met behulp van de Docker afbeelding en de descriptor file package.yaml

2 - Exporteer de afbeelding als een root-bestandssysteem en pak deze in met het descriptor YAML-bestand

Optie 1 – De Docker-afbeelding en het YAML-bestand verpakken:

Ga naar de doelmap waar u de afbeelding en het YAML-bestand wilt verpakken.

```
ubuntu@ip-172-31-30-249:~/tes0$ ls
package.yaml
```

Pak het bestand vervolgens in door deze opdracht uit te voeren:

```
ioxclient docker package tac_app package.yaml
```

```
...

Example:
ubuntu@ip-172-31-30-249:~/tese$ sudo /home/ubuntu/ioxclient_1.17.0.0_linux_amd64/ioxclient docker packa
Currently active profile: default
Secure client authentication: no
Command Name: docker-package
Timestamp at DockerPackage start: 1748211382584
Using the package descriptor file in the project dir
Validating descriptor file package.yaml with package schema definitions
Parsing descriptor file..
Found schema version 2.7
Loading schema file for version 2.7
Validating package descriptor file..
File package.yaml is valid under schema version 2.7
Generating 10x package of type docker with layers as rootfs
Replacing symbolically linked layers in docker rootfs, if any
No symbolically linked layers found in rootfs. No changes made in rootfs
Removing emulation layers in docker rootfs, if any
The docker image is better left in it's pristine state
Updated package metadata file :/home/ubuntu/tes0/.package.metadata
No rsa key and/or certificate files provided to sign the package
------------------------------------------------------------------
Generating the envelope package
------------------------------------------------------------------
Checking if package descriptor file is present..
Skipping descriptor schema validation..
Created Staging directory at : /tmp/1093485025
Copying contents to staging directory
Timestamp before CopyTree: 1748211503878
Timestamp after CopyTree: 1748211575671
Creating artifacts manifest file
Creating an inner envelope for application artifacts
Including rootfs.tar
Generated /tmp/1093485025/artifacts.tar.gz
Parsing Package Metadata file /tmp/1093485025/.package.metadata
Updated package metadata file /tmp/1093485025/.package.metadata
Calculating SHA256 checksum for package contents..
Timestamp before SHA256: 1748211630718
Timestamp after SHA256: 1748211630718
Path:.package.metadata
SHA256: 50c922f103ddc01a5dc7a98d6cacefb167f4a2c692dfc521231bb42f0c3dcf55 Timestamp before SHA256: 17482
Timestamp after SHA256: 1748211630719
Path: artifacts.mf
SHA256: 51100 8aa2d1418daf1770768fb79c90f16814ff7789d03beb4f4ea1bf4fae8f2 Timestamp before SHA256: 17482
Timestamp after SHA256: 1748211634941
Path: artifacts.tar.gz
SHA256: 0cc3f69af50cf0a01ec9a1400c440f60a0dff55369bd309b6dfc69715302425+ Timestamp before SHA256: 17482
Timestamp after SHA256: 1748211634952
Path: envelope_package.tar.gz
SHA256: d492de09441a241f879cd268cd1b3424ee79a58a9495aa77ae5b11cab2fd55da Timestamp before SHA256: 17482
Timestamp after SHA256: 1748211634963
Path: package.yaml
SHA256: d8dc7253443ff3ad080c42bc8d82db4c3ea7ae9b2d0e2f827fbaf2bc80245f62 Generated package manifest at
Generating IOx Package..
Package docker image tac_app at /home/ubuntu/tes0/package.tar
ubuntu@ip-172-31-30-249:~/tes0$ |
```

Deze reeks acties was verantwoordelijk voor de productie van het pakket teerbundel. Om de

inhoud van het pakket te inspecteren, kunnen we het decomprimeren met behulp van het teer-hulpprogramma

```
ubuntu@ip-172-31-30-249:~/tes0$ tar -tf package.tar
package.yaml
artifacts.mf
.package.metadata
package.mf
envelope_package.tar.gz
artifacts.tar.gz
```

Optie 2 - De Docker-afbeelding exporteren als een rootbestandssysteem en deze verpakken met het beschrijvende YAML-bestand.

Voer de opdracht uit in de map waarin het image moet worden gemaakt:

```
ubuntu@ip-172-31-30-249:~/tac_app$ sudo docker save tac_app -o rootfs.tar
```

Met deze opdracht wordt het Docker-image geëxporteerd als een bundel met het rootbestandssysteem dat op / in de container wordt gemonteerd.

Verplaats het bestand package.YAML naar de opgegeven locatie. Na voltooiing moet de directorystructuur er als volgt uitzien:

```
ubuntu@ip-172-31-30-249:~/tac_app$ ls
package.yaml   rootfs.tar
```

De laatste stap bestaat uit het verpakken van de Docker-afbeelding door deze opdracht uit te voeren:

```
ioxclient docker package tac_app package.yaml
...

ubuntu@ip-172-31-30-249:~/tac_app$ ioxclient package .
Currently active profile :  default
Secure client authentication:  no
Command Name:  package
No rsa key and/or certificate files provided to sign the package
Checking if package descriptor file is present..
Validating descriptor file /home/ubuntu/tac_app/package.yaml with package schema definitions
Parsing descriptor file..
Found schema version  2.7
Loading schema file for version  2.7
Validating package descriptor file..
```

```
File /home/ubuntu/tac_app/package.yaml is valid under schema version 2.7
Created Staging directory at :  /tmp/2119895371
Copying contents to staging directory
Timestamp before CopyTree: 1748374177879


Timestamp after CopyTree: 1748374357306
Creating artifacts manifest file
Creating an inner envelope for application artifacts

Generated  /tmp/2119895371/artifacts.tar.gz
Updated package metadata file :  /tmp/2119895371/.package.metadata
Calculating SHA256 checksum for package contents..
Timestamp before SHA256: 1748374566796
Timestamp after SHA256: 1748374566796
Path:  .package.metadata
SHA256 : 4fad07c3ac4d817db17bacc8563b4c632bc408d2a9cbdcb5e7a526c1c5c6e04e
Timestamp before SHA256: 1748374566796
Timestamp after SHA256: 1748374566809
Path:  artifacts.mf
SHA256 : d448a678ae952f9fe74dc19172aba17e283a5e268aca817fefc78b585f02b492
Timestamp before SHA256: 1748374566809
Timestamp after SHA256: 1748374575477
Path:  artifacts.tar.gz
SHA256 : 64d70f43be692e3cee61d906036efef45ba29e945437237e1870628ce64d5147
Timestamp before SHA256: 1748374575477
Timestamp after SHA256: 1748374575489
Path:  package.yaml
SHA256 : d8dc7253443ff3ad080c42bc8d82db4c3ea7ae9b2d0e2f827fbaf2bc80245f62
Generated package manifest at  package.mf
Generating IOx Package..
Package generated at /home/ubuntu/tac_app/package.tar
```

Als gevolg van deze acties wordt het bestand package.tar gegenereerd en voorbereid voor implementatie. Voer de volgende opdracht uit om de inhoud van het pakket te bekijken:

```
ubuntu@ip-172-31-30-249:~/tac_app$ tar -tf  tac_app.tar
package.yaml
artifacts.mf
.package.metadata
package.mf
artifacts.tar.gz
```

Installatie

Nadat de toepassing is voorbereid, bestaat de laatste stap uit het installeren op het doelapparaat door de opdracht in de geprivilegieerde EXEC-modus uit te voeren, zoals wordt weergegeven:

```
 app-hosting install appid tacapp package bootflash:package.tar
```

Wacht ongeveer 1 minuut en bevestig of de toepassing succesvol wordt uitgevoerd:

```
Router# show app-hosting list
App id                                   State
---------------------------------------------------------
  tacapp                                 RUNNING
```

Over deze vertaling

Cisco heeft dit document vertaald via een combinatie van machine- en menselijke technologie om onze gebruikers wereldwijd ondersteuningscontent te bieden in hun eigen taal. Houd er rekening mee dat zelfs de beste machinevertaling niet net zo nauwkeurig is als die van een professionele vertaler. Cisco Systems, Inc. is niet aansprakelijk voor de nauwkeurigheid van deze vertalingen en raadt aan altijd het oorspronkelijke Engelstalige document (link) te raadplegen.