

Opzetten en implementeren van een Docker IOx-pakket voor de IR1101 ARM-architectuur

Inhoud

[Inleiding](#)

[Voorwaarden](#)

[Vereisten](#)

[Gebruikte componenten](#)

[Achtergrondinformatie](#)

[Configureren](#)

[Deel 1. Bouw het IOx-pakket voor IR1101](#)

[1. Installeer en bereid IOx-client op de Linux-host](#)

[2. Installeer en bereid de Docker Environment op de Linux Build Machine](#)

[3. Installeer de QEMU-gebruikersemulatiepakketten](#)

[4. Test of een Aken64/ARV64v8-container op x86 Linux-machine draait](#)

[5. Bereid bestanden voor om de Docker Webserver-container te maken](#)

[6. Bouw de Docker-container](#)

[7. Het IOx-pakket maken](#)

[Deel 2. Instellen van de IR1101 voor IOx](#)

[1. Schakel de Webinterface, IOx en Local Manager in](#)

[2. IOx-netwerken configureren](#)

[Deel 3. Toegang tot lokale Manager en implementatie van de IOx-toepassing](#)

[Verifiëren](#)

[Problemen oplossen](#)

Inleiding

Dit document beschrijft hoe u een Docker-gebaseerd IOx-pakket kunt voorbereiden, bouwen en implementeren voor de IR1101 ARM-gebaseerde Internet of Things (IoT)-poort.

Voorwaarden

Vereisten

Cisco raadt kennis van de volgende onderwerpen aan:

- Linux
- containers
- IOx

Gebruikte componenten

De informatie in dit document is gebaseerd op de volgende software- en hardware-versies:

- IR1101 dat via Secure Shell (SSH) bereikbaar is
IP-adres ingesteld Toegang tot het apparaat met een recht van 15 gebruikers
- Linux host (een minimale Debian 9 (stretch) installatie wordt gebruikt voor dit artikel)
- IOx-clientinstallatiebestanden die kunnen worden gedownload van:
<https://software.cisco.com/download/release.html?mdfid=286306005&softwareid=286306762>

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 de potentiële impact van elke opdracht begrijpen.

Achtergrondinformatie

De IR1101 is een beetje anders in vergelijking met de meeste andere IOx-platforms, omdat deze voornamelijk op x86 zijn gebaseerd. De IR1101 is gebaseerd op de ARM64v8 architectuur zodat u geen containers of IOx pakketten kunt inzetten die voor x86 op het platform rechtstreeks zijn gebouwd. Dit document begint bij nul en bereidt de omgeving voor om ARM64v8-gebaseerde Docker containers te bouwen en legt uit hoe ze op de IR1101 kunnen bouwen, verpakken en inzetten met behulp van een x86 PC.

Een zeer klein Python-schrift dat een eenvoudige webserver is, wordt gebruikt en er wordt een Docker-container gebouwd om het uiteindelijk te verpakken om het op de IR1101 te laten draaien. Het enige wat de webserver zal doen is luisteren op een vooraf gedefinieerde poort (9000) en een simpele pagina teruggeven wanneer het een **GET** aanvraag ontvangt. Hiermee kunt u de mogelijkheid testen om uw eigen code uit te voeren en kunt u de netwerktoegang tot de IOx-toepassing testen zodra deze start.

Het pakket wordt gebouwd door de Docker-gereedschappen, met behulp van Alpine Linux. Alpine Linux is een kleine Linux-afbeelding (ongeveer 5MB), die vaak wordt gebruikt als basis voor Docker-containers.

Aangezien de meeste desktop/laptop/VM's in de buurt allemaal op x86 gebaseerd zijn, moet u de ARM64v8 architectuur nastreven op de op x86 gebaseerde machine waar de container gebouwd is. U kunt dit gemakkelijk doen door het gebruik van Quick Emulator (QEMU) - gebruikersemulatie. Dit maakt de uitvoering van uitvoerbare objecten in een niet-inheemse architectuur mogelijk, net zoals het op zijn eigen architectuur zou draaien.

Configureren

Deel 1. Bouw het IOx-pakket voor IR1101

1. Installeer en bereid IOx-client op de Linux-host

Je hebt `ioxclient` nodig om de Docker-container te verpakken als een IOx-pakket zodra het gebouwd is, dus laten we dit eerst voorbereiden.

Eerste kopie of download het `ioxclient`-pakket. Het is beschikbaar op:

<https://software.cisco.com/download/release.html?mdfid=286306005&softwareid=286306762>.

```
.
jedepuyd@192.168.56.101's password:
ioxclient_1.7.0.0_linux_amd64.tar.gz
```

```
100% 4798KB 75.2MB/s 00:00
```

De verpakking opzuigen:

```
jedepuyd@deb9:~$ tar -xvzf ioxclient_1.7.0.0_linux_amd64.tar.gz
ioxclient_1.7.0.0_linux_amd64/ioxclient
ioxclient_1.7.0.0_linux_amd64/README.md
```

Voeg het pad toe aan de **PATH** variabele om het beschikbaar te hebben zonder het gebruik van de volledige plaats. Als u de machine opnieuw start of een schakelaar aansluit op de gebruikers, vergeet dan niet deze stap te herhalen:

```
jedepuyd@deb9:~$ export PATH=$PATH:/home/jedepuyd/ioxclient_1.7.0.0_linux_amd64/
```

Start **ioxclient** voor het eerst om een verplicht profiel te maken. Aangezien u **ioxclient** alleen gebruikt om de Docker-container te verpakken, kunnen de waarden standaard worden gelaten:

```
jedepuyd@deb9:~$ ioxclient -v
ioxclient version 1.7.0.0
jedepuyd@deb9:~/iox_aarch64_webserver$ ioxclient profiles reset
Active Profile: default
Your current config details will be lost. Continue (y/N) ? : y
Current config backed up at /tmp/ioxclient731611124
Config data deleted.
jedepuyd@deb9:~/iox_aarch64_webserver$ ioxclient -v
Config file not found : /home/jedepuyd/.ioxclientcfg.yaml
Creating one time configuration..
Your / your organization's name :
Your / your organization's URL :
Your IOx platform's IP address[127.0.0.1] :
Your IOx platform's port number[8443] :
Authorized user name[root] :
Password for root :
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]:
Your RSA key, for signing packages, in PEM format[:]:
Your x.509 certificate in PEM format[:]:
Activating Profile default
Saving current configuration
ioxclient version 1.7.0.0
```

2. Installeer en bereid de Docker Environment op de Linux Build Machine

Deze Docker wordt gebruikt om een container van het Alpenbasebeeld te bouwen en om de benodigde bestanden voor de gebruikscase op te nemen. De gegeven stappen zijn gebaseerd op de officiële installatiegidsen van Docker Community Edition (CE) voor Debian:

<https://docs.docker.com/install/linux/docker-ce/debian/>

update de pakketlijsten op uw machine:

```
jedepuyd@deb9:~$ sudo apt-get update
...
```

```
Reading package lists... Done
```

Installeer de gebiedsdelen zodanig dat u de Docker-melding kunt gebruiken:

```
jedepuyd@deb9:~$ sudo apt-get install apt-transport-https ca-certificates curl gnupg2 software-properties-common
Reading package lists... Done
Building dependency tree
...
Processing triggers for dbus (1.10.26-0+deb9u1) ...
```

Voeg de Docker GNU Privacy Guard (GPG) toets toe als een geldige GPG-toets:

```
jedepuyd@deb9:~$ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -
OK
```

Controleer de vingerafdruk van de geïnstalleerde GPG-toets:

```
jedepuyd@deb9:~$ sudo apt-key fingerprint 0EBFCD88
pub  rsa4096 2017-02-22 [SCEA]
     9DC8 5822 9FC7 DD38 854A  E2D8 8D81 803C 0EBF CD88
uid          [ unknown] Docker Release (CE deb) <docker@docker.com>
sub  rsa4096 2017-02-22 [S]
```

Docker stabiel rapport toevoegen:

```
jedepuyd@deb9:~$ sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/debian $(lsb_release -cs) stable"
update de pakketlijsten opnieuw terwijl u de Docker Repos toevoegt:
```

```
jedepuyd@deb9:~$ sudo apt-get update
...
Reading package lists... Done
```

Docker installeren:

```
jedepuyd@deb9:~$ sudo apt-get install docker-ce docker-ce-cli containerd.io
Reading package lists... Done
Building dependency tree
...
Processing triggers for systemd (232-25+deb9u9) ...
```

U kunt Docker als een normale gebruiker benaderen/uitvoeren door deze gebruiker aan de Docker-groep toe te voegen en het lidmaatschap van de groep op te frissen:

```
jedepuyd@deb9:~$ sudo usermod -a -G docker jedepuyd
jedepuyd@deb9:~$ newgrp docker
```

3. Installeer de QEMU-gebruikersemulatiepakketten

Nadat u Docker heeft geïnstalleerd, moet u de QEMU-gebruikersemulators installeren. Gebruik de statisch verbonden QEMU emulator vanuit de Docker-container zodat u de container voor ARM64v8 kunt draaien op onze x86-gebaseerde Linux-machine, hoewel de doelcontainer ontworpen is voor de ARM64v8 architectuur.

Installeer de verpakkingen:

```
jedepuyd@deb9:~$ sudo apt-get install qemu-user qemu-user-static
```

```
Reading package lists... Done
Building dependency tree
...
Processing triggers for man-db (2.7.6.1-2) ...
```

Na de installatie zijn hier de statistisch verbonden QEMU-emulators beschikbaar in **usr/bin**:

```
jedepuyd@deb9:~$ ls -al /usr/bin/qemu-*static
-rwxr-xr-x 1 root root 3468784 Nov  8 16:41 /usr/bin/qemu-aarch64-static
-rwxr-xr-x 1 root root 2791408 Nov  8 16:41 /usr/bin/qemu-alpha-static
-rwxr-xr-x 1 root root 3399344 Nov  8 16:41 /usr/bin/qemu-armeb-static
-rwxr-xr-x 1 root root 3391152 Nov  8 16:41 /usr/bin/qemu-arm-static
-rwxr-xr-x 1 root root 2800400 Nov  8 16:41 /usr/bin/qemu-cris-static
...
```

Het eerste in de lijst is wat je nodig hebt: aarch64 is de achternaam van ARM64v8 voor Linux.

4. Test of een Aken64/ARV64v8-container op x86 Linux-machine draait

Nu Docker is geïnstalleerd en de benodigde binaire QEMU binaries zijn geïnstalleerd, kunt u testen of u een Docker container kunt draaien die voor ARM64v8 op de x86 machine is gebouwd:

```
jedepuyd@deb9:~$ docker run -v /usr/bin/qemu-aarch64-static:/usr/bin/qemu-aarch64-static --rm -
ti arm64v8/alpine:3.7
Unable to find image 'arm64v8/alpine:3.7' locally
3.7: Pulling from arm64v8/alpine
40223db5366f: Pull complete
Digest: sha256:a50c0cd3b41129046184591963a7a7682277736258e5ade8445b07c88bfdcc3
Status: Downloaded newer image for arm64v8/alpine:3.7
/ # uname -a
Linux 1dbba69b60c5 4.9.0-8-amd64 #1 SMP Debian 4.9.144-3.1 (2019-02-19) aarch64 Linux
```

Zoals u kunt zien in de output, wordt arm64v8 Alpencontainer verkregen en gemaakt om te lopen met toegang tot de emulator.

Als je de architectuur van de container vraagt, kan je zien dat de code gecompileerd is voor maart 64. Precies zoals de doelboom voor de container zou moeten zijn voor IR1101.

5. Bereid bestanden voor om de Docker Webserver-container te maken

Nu alle voorbereiding klaar is, kun je de benodigde bestanden maken voor de webservercontainer die gebruikt moet worden op IR1101.

Het eerste bestand is **webserver.py**, het Python-script dat je in de container wilt gebruiken. Aangezien dit slechts een voorbeeld is, kunt u dit uiteraard vervangen door de eigenlijke code om in uw IOx-toepassing te kunnen uitvoeren:

```
jedepuyd@deb9:~$ mkdir iox_aarch64_webserver
jedepuyd@deb9:~$ cd iox_aarch64_webserver

jedepuyd@deb9:~/iox_aarch64_webserver$ vi webserver.py
jedepuyd@deb9:~/iox_aarch64_webserver$ 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 on arm64v8</h1></body></html>")
        logf.write('Got GET\n')
        logf.flush()

def run(server_class=HTTPServer, handler_class=S, port=9000):
    server_address = ('', port)
    httpd = server_class(server_address, handler_class)
    print 'Starting webserver...'
    logf.write('Starting webserver...\n')
    logf.flush()
    httpd.serve_forever()

if __name__ == "__main__":
    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')
    run()
    logf.close()

```

Deze code bevat de logica om naar een logbestand te schrijven, dat beschikbaar is voor raadpleging van de lokale beheerder.

Het tweede bestand dat nodig is, is het Dockerfile. Dit definieert hoe de container gebouwd is:

```

jedepuyd@deb9:~/iox_aarch64_webserver$ vi Dockerfile
jedepuyd@deb9:~/iox_aarch64_webserver$ cat Dockerfile
FROM arm64v8/alpine:3.7
COPY qemu-aarch64-static /usr/bin

RUN apk add --no-cache python
COPY webserver.py /webserver.py

```

Het Dockerfile bepaalt hoe de container gebouwd zal worden. Start vanaf de Alpine-basisafbeelding voor ARM64v8, kopieer de emulator in de verpakking, voer de apk uit om het Python-pakket toe te voegen en kopieer het webserver-script in de container.

Laatste voorbereiding die nodig is voordat je de container kunt bouwen is het kopiëren van qemu-aarch64-statisch naar de folder vanaf waar je de container bouwt:

```

jedepuyd@deb9:~/iox_aarch64_webserver$ cp /usr/bin/qemu-aarch64-static .

```

6. Bouw de Docker-container

Nu alle voorbereiding klaar is, kunt u de container bouwen met het gebruik van Dockerfile:

```

jedepuyd@deb9:~/iox_aarch64_webserver$ docker build -t iox_aarch64_webserver .
Sending build context to Docker daemon 3.473MB
Step 1/4 : FROM arm64v8/alpine:3.7
----> e013d5426294
Step 2/4 : COPY qemu-aarch64-static /usr/bin

```

```

---> addf4e1cc965
Step 3/4 : RUN apk add --no-cache python
---> Running in ff3768926645
fetch http://dl-cdn.alpinelinux.org/alpine/v3.7/main/aarch64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.7/community/aarch64/APKINDEX.tar.gz
(1/10) Installing libbz2 (1.0.6-r6)
(2/10) Installing expat (2.2.5-r0)
(3/10) Installing libffi (3.2.1-r4)
(4/10) Installing gdbm (1.13-r1)
(5/10) Installing ncurses-terminfo-base (6.0_p20171125-r1)
(6/10) Installing ncurses-terminfo (6.0_p20171125-r1)
(7/10) Installing ncurses-libs (6.0_p20171125-r1)
(8/10) Installing readline (7.0.003-r0)
(9/10) Installing sqlite-libs (3.25.3-r0)
(10/10) Installing python2 (2.7.15-r2)
Executing busybox-1.27.2-r11.trigger
OK: 51 MiB in 23 packages
Removing intermediate container ff3768926645
---> eda469dab9c6
Step 4/4 : COPY webserver.py /webserver.py
---> ccf7ee7227c9
Successfully built ccf7ee7227c9
Successfully tagged iox_aarch64_webserver:latest

```

Als test, gebruik de container die u net bouwde en controleer of het script werkt:

```

jedepuyd@deb9:~/iox_aarch64_webserver$ docker run -ti iox_aarch64_webserver
/ # uname -a
Linux dae047f1a6b2 4.9.0-8-amd64 #1 SMP Debian 4.9.144-3.1 (2019-02-19) aarch64 Linux
/ # python webserver.py &
/ # Starting webserver...

/ # netstat -tlnp
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      13/qemu-aarch64-
sta
/ # exit

```

Zoals je kunt zien in deze output, is de architectuur van de container de geëngageerde arch64. Nadat je het script start, zie je dat het luistert naar verzoeken op poort 9000.

7. Het IOx-pakket maken

De verpakking is klaar om te worden verpakt. Voordat u ioxclient kunt vragen dit te doen, moet u eerst de pakketbeschrijver maken: **verpakking.yaml**.

Dit bestand beschrijft hoe het pakket eruit zou moeten zien, hoeveel bronnen het nodig heeft om te starten en hoe het eruit zou moeten zien.

```

jedepuyd@deb9:~/iox_aarch64_webserver$ vi package.yaml
jedepuyd@deb9:~/iox_aarch64_webserver$ cat package.yaml
descriptor-schema-version: "2.7"

info:
  name: "iox_aarch64_webserver"
  description: "simple docker webserver for arm64v8"
  version: "1.0"
  author-link: "http://www.cisco.com"
  author-name: "Jens Depuydt"

```

```

app:
  cpuarch: "aarch64"
  type: "docker"
  resources:
    profile: cl.tiny
    network:
      -
        interface-name: eth0
        ports:
          tcp: ["9000"]

  startup:
    rootfs: rootfs.tar
    target: ["python", "/webserver.py"]

```

Zoals u kunt zien, is de CPU-architectuur ingesteld op arch64. Om toegang tot TCP-poort 9000 te krijgen, kunt u **rootfs.tar** gebruiken als de poorten en aan het begin van de taak, **python/webserver.py** uitvoeren.

Het laatste wat je moet doen voordat je kan verpakken is de **rootfs.tar** uit de Docker container halen:

```
jedepuyd@deb9:~/iox_aarch64_webserver$ docker save -o rootfs.tar iox_aarch64_webserver
```

Op dit punt kunt u **ioxclient** gebruiken om het IOx-pakket voor IR1101 te bouwen:

```

jedepuyd@deb9:~/iox_aarch64_webserver$ ioxclient package .
Currently active profile : default
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/jedepuyd/iox_aarch64_webserver/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/jedepuyd/iox_aarch64_webserver/package.yaml is valid under schema version 2.7
Created Staging directory at : /tmp/017226485
Copying contents to staging directory
Creating an inner envelope for application artifacts
Generated /tmp/017226485/artifacts.tar.gz
Calculating SHA1 checksum for package contents..
Updated package metadata file : /tmp/017226485/.package.metadata
Root Directory : /tmp/017226485
Output file: /tmp/475248592
Path: .package.metadata
SHA1 : 95abe28fc05395fc5f71f7c28f59eceb1495bf9b
Path: artifacts.tar.gz
SHA1 : bdf5596a0747eae51bb0ald2870fd09a5a16a098
Path: package.yaml
SHA1 : e65a6fcbe96725dd5a09b60036448106acc0c138
Generated package manifest at package.mf
Generating IOx Package..
Package generated at /home/jedepuyd/iox_aarch64_webserver/package.tar

```

Op dit moment is er een pakket om op de IR1101 klaar te zetten als pakket.tar. Het volgende deel legt uit hoe het apparaat voor plaatsing moet worden voorbereid.

Deel 2. Instellen van de IR1101 voor IOx

1. Schakel de Webinterface, IOx en Local Manager in

Local Manager is een GUI om IOx-toepassingen te implementeren, activeren, starten, beheren en probleemoplossing. Voor IR1101 is het ingebed in de reguliere management web interface. Dus moet je dat eerst mogelijk maken.

Voer deze stappen uit op de IR1101 om IOx en de web interface in te schakelen.

```
BRU_IR1101_20#conf t
Enter configuration commands, one per line. End with CNTL/Z.
BRU_IR1101_20(config)#iox
BRU_IR1101_20(config)#ip http server
BRU_IR1101_20(config)#ip http secure-server
BRU_IR1101_20(config)#ip http authentication local
BRU_IR1101_20(config)#username admin privilege 15 password 0 cisco
```

De laatste regel voegt een gebruiker toe met 15 voorrecht. Deze gebruiker heeft toegang tot de webinterface en IOx-beheerder.

2. IOx-netwerken configureren

Voordat u tot de web interface toegang hebt, moeten we de gewenste configuratie voor het IOx-netwerk toevoegen. Achtergrondinformatie is te vinden in de IR1101 documentatie voor IOx: https://www.cisco.com/c/en/us/td/docs/routers/access/1101/software/configuration/guide/b_IR1101_config/b_IR1101config_chapter_010001.html

In het kort kunnen de IOx-toepassingen met de buitenwereld communiceren met het gebruik van de VirtualPortGroup0-interface (vergelijkbaar met de Gi2 op IR809 en Gi5 op IR829 interfaces).

```
BRU_IR1101_20(config)#interface VirtualPortGroup0
BRU_IR1101_20(config-if)# ip address 192.168.1.1 255.255.255.0
BRU_IR1101_20(config-if)# ip nat inside
BRU_IR1101_20(config-if)# ip virtual-reassembly
BRU_IR1101_20(config-if)#exit
```

Aangezien u de VirtualPortGroup0-interface configureren als netwerkadresomzetting (NAT) binnen, moet u de ip-ingang buiten statement op de Gi 0/0/0 interface toevoegen om communicatie naar en van de IOx-toepassingen met behulp van NAT toe te staan:

```
BRU_IR1101_20(config)#interface gigabitEthernet 0/0/0
BRU_IR1101_20(config-if)#ip nat outside
BRU_IR1101_20(config-if)#ip virtual-reassembly
```

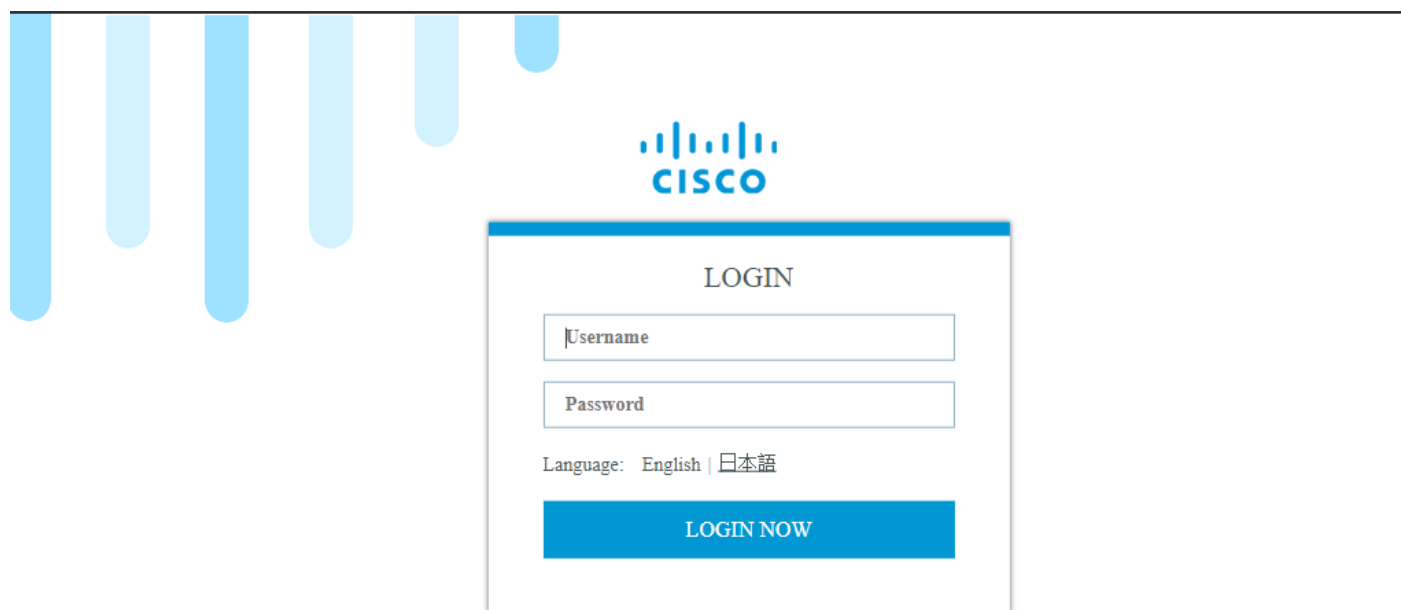
Om toegang tot haven 9000 voor de container mogelijk te maken, die u 192.168.1.15 kunt geven, moet u een haven voorwaarts toevoegen:

```
BRU_IR1101_20(config)#ip nat inside source static tcp 192.168.1.15 9000 interface
GigabitEthernet0/0/0 9000
```

Gebruik voor deze handleiding statisch geconfigureerd IP's per IOx-toepassing. Als u IP-adressen dynamisch aan de toepassingen wilt toewijzen, moet u de configuratie voor een DHCP-server in het subtype VirtualPortGroup0 toevoegen.

Deel 3. Toegang tot lokale Manager en implementatie van de IOx-toepassing

Nadat u deze lijnen aan de configuratie toevoegt, kunt u de IR1101 gebruiken met de web interface. Navigeer naar het Gi 0/0/0 IP adres met het gebruik van uw browser zoals getoond in de afbeelding.



© 2005-2018 - Cisco Systems, Inc. All rights reserved. Cisco, the Cisco logo, and Cisco Systems are registered trademarks or trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries. All third party trademarks are the property of their respective owners.

Gebruik de bevoorrechte 15-account die in Stap 1 is gemaakt, om in te loggen op de webinterface en in te navigeren naar **Configuration** - IOx zoals in de afbeelding.



Search Menu Items

Dashboard

Monitoring >

Configuration >

Administration >

Troubleshooting

Interface

Cellular

Ethernet

Logical

Layer2

VLAN

VTP

Routing Protocols

EIGRP

OSPF

Static Routing

Security

AAA

ACL

NAT

VPN

Services

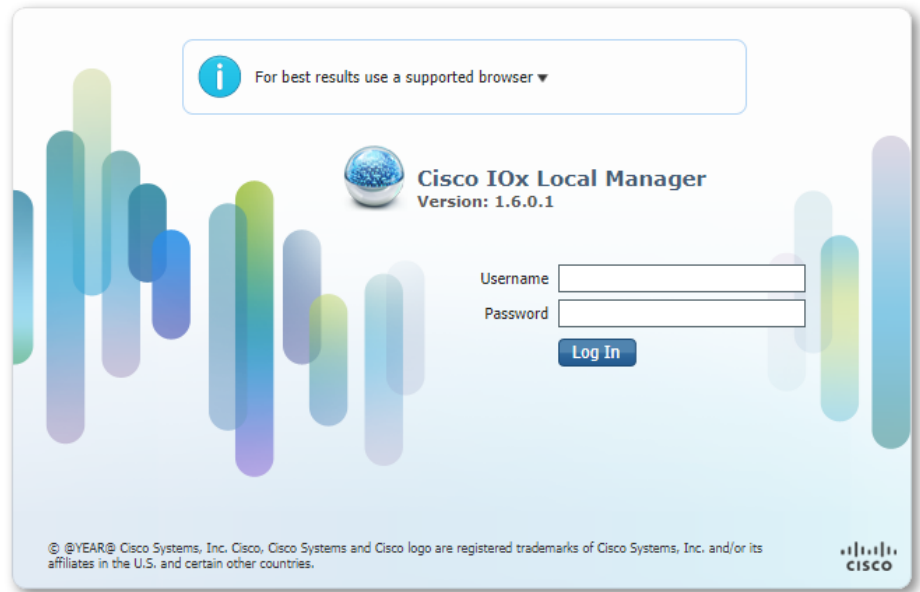
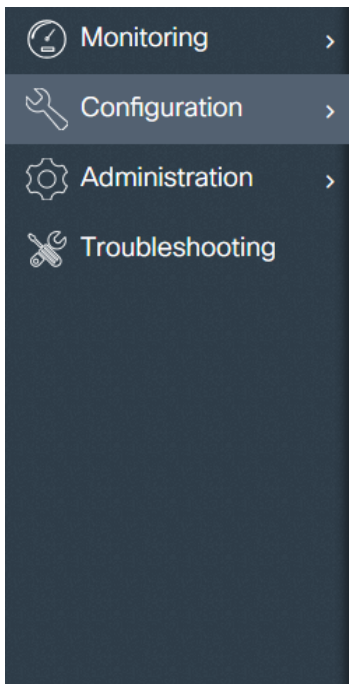
Application Visibility

Custom Application

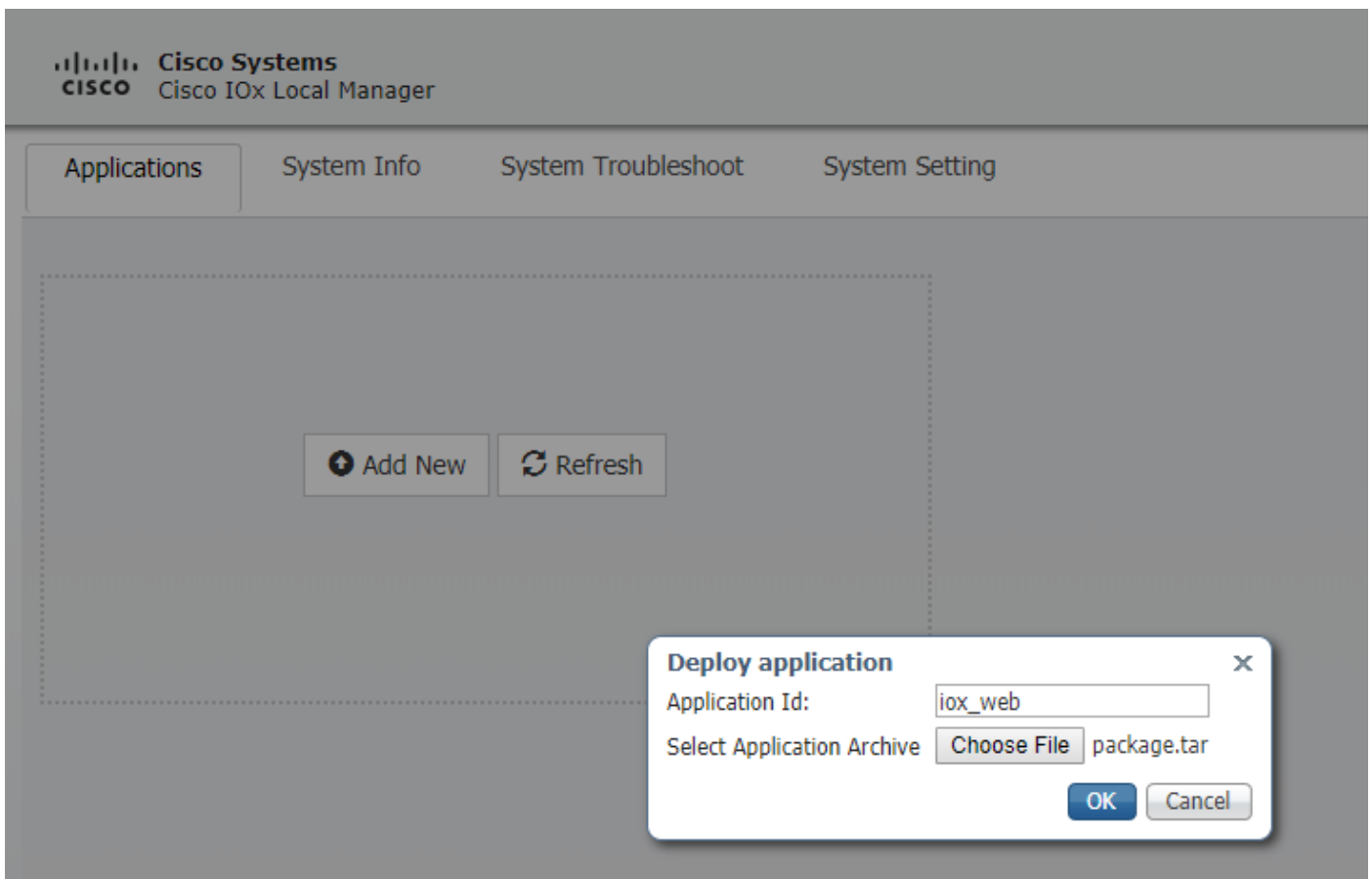
IOx

NETFLOW

In de inlognaam van IOx Local Manager gebruikt u dezelfde account om verder te gaan zoals in de afbeelding.



Klik op **Add New**, selecteer een naam voor de IOx-toepassing en kies het pakket.tar dat in Deel 1 is gebouwd zoals in de afbeelding.



Nadat het pakket is geüpload, kunt u het activeren zoals in de afbeelding wordt weergegeven.

iox_web

DEPLOYED

simple docker webserver for arm64v8

TYPE	VERSION	PROFILE
docker	1.0	c1.tiny

Memory *	6.3%
----------	------

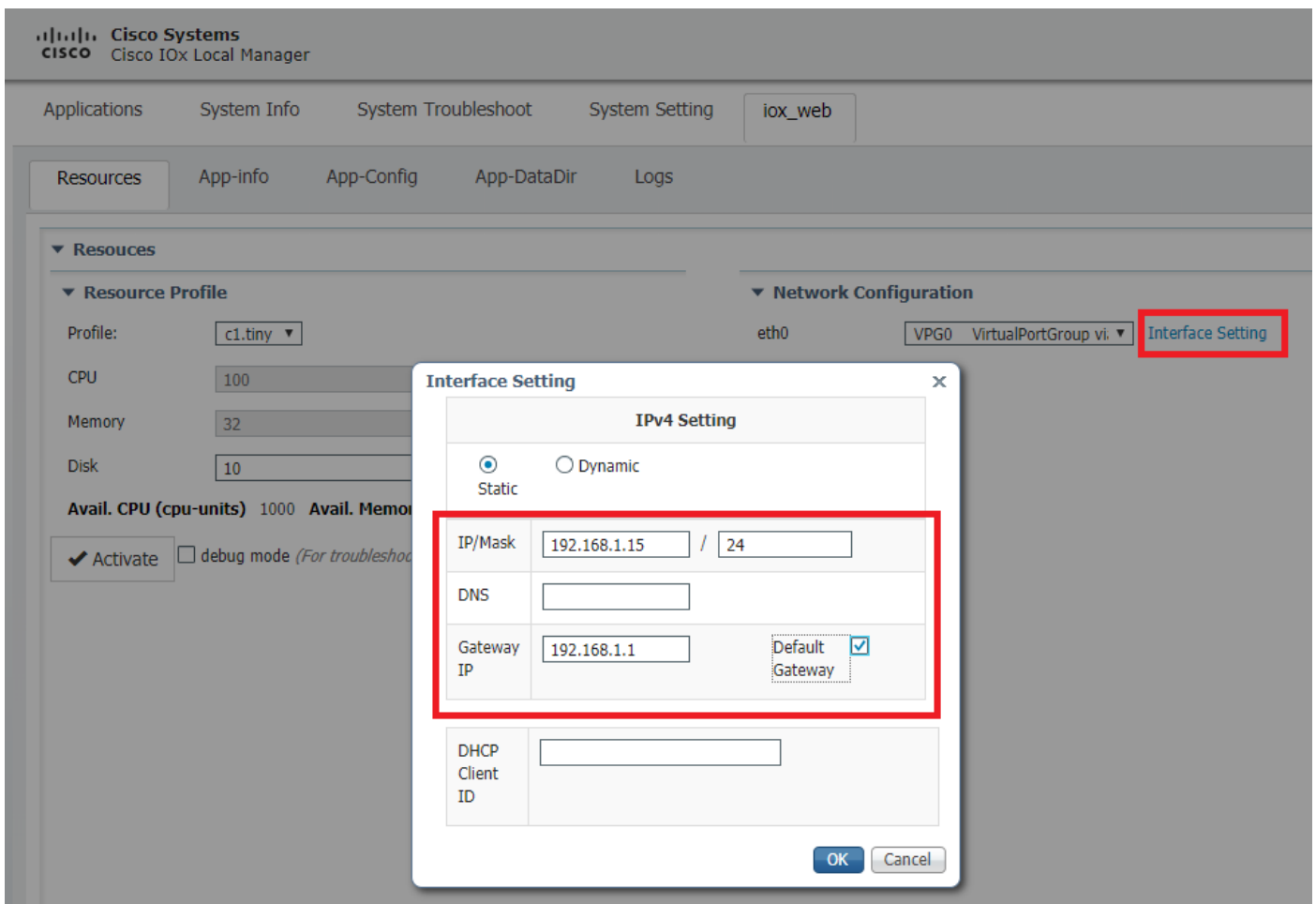
CPU *	10.0%
-------	-------

✓ Activate

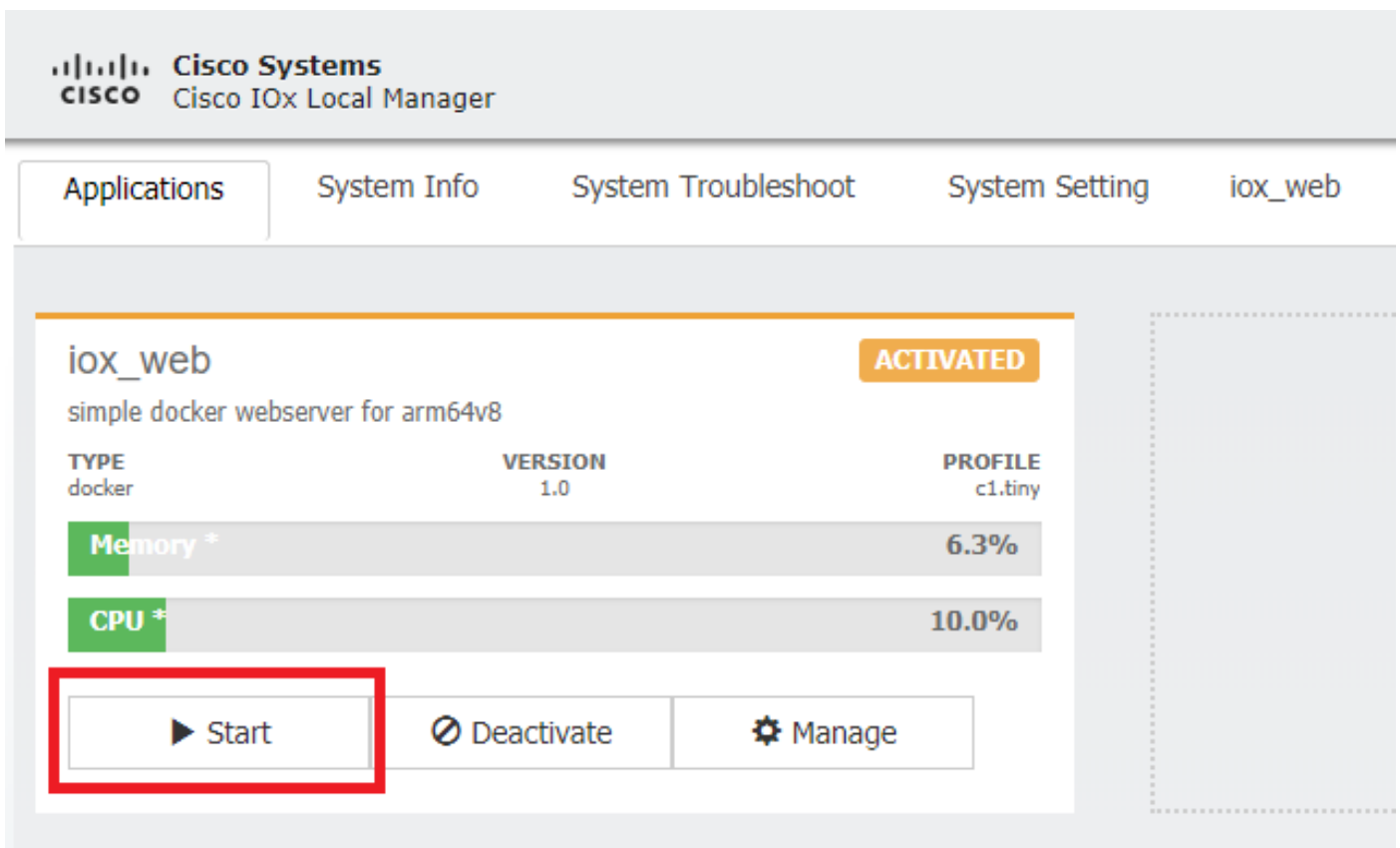
Upgrade

Delete

In het tabblad **Resources** opent u de interface-instelling om de vaste IP te specificeren die u aan de app wilt toewijzen zoals in de afbeelding.



Klik op **OK** en **activeer** vervolgens **het programma**. Nadat de actie is voltooid, navigeer dan terug naar de hoofdpagina voor Local Manager (de knop **Toepassingen** in het bovenste menu) en start de toepassing zoals in de afbeelding.



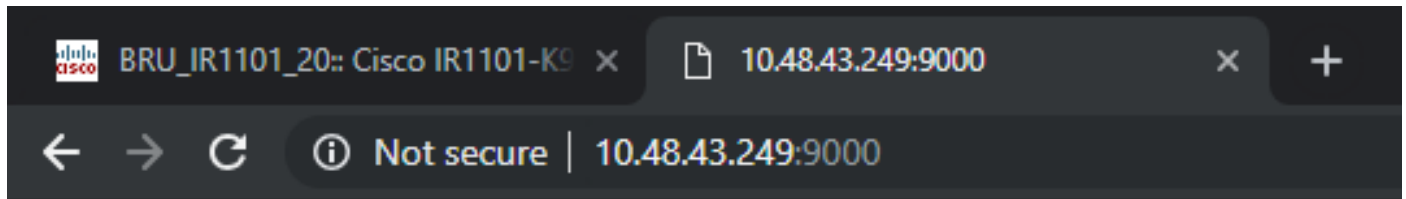
Nadat u deze stappen heeft doorlopen, dient uw applicatie te starten en beschikbaar te zijn via poort 9000 met gebruik van de Gi 0/0/0 interface van de IR1101.

Verifiëren

Gebruik dit gedeelte om te bevestigen dat de configuratie correct werkt.

Om te verifiëren kunt u tot het IP adres van de Gi 0/0/0 interface op de IR1101 met het gebruik van haven 9000 toegang hebben.

Als alles goed gaat, zou je dit als volgt moeten zien, zoals het in het Python-schrift is gemaakt.



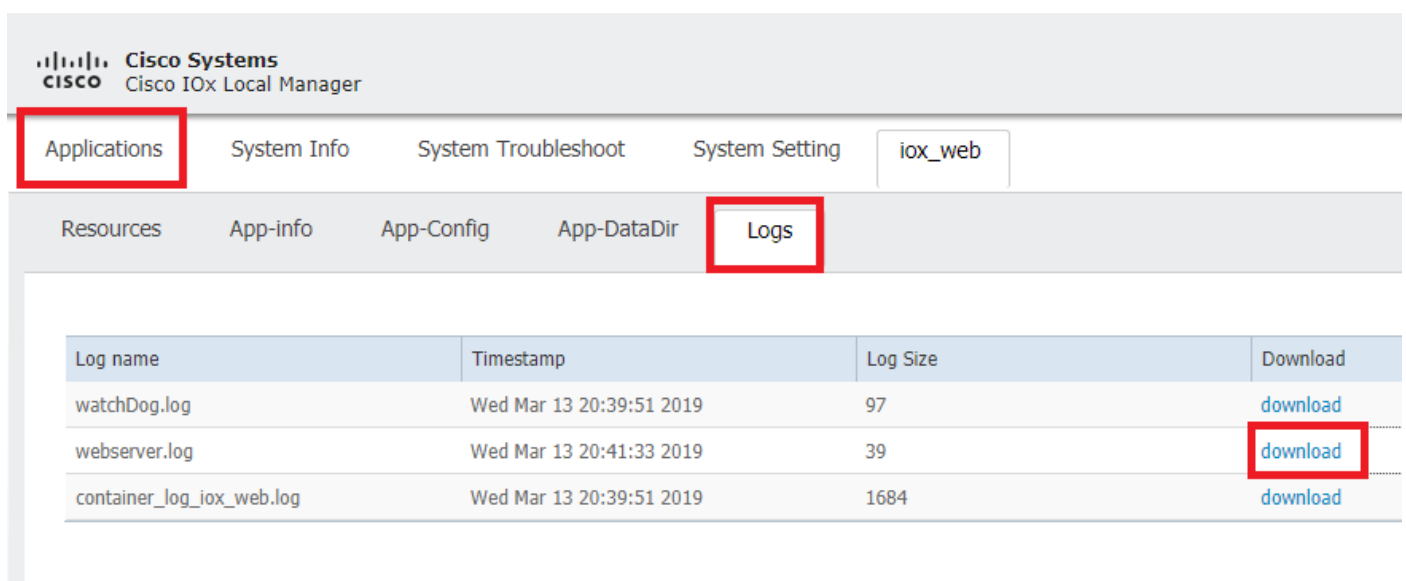
IOX python webserver on arm64v8

Problemen oplossen

Deze sectie verschaft informatie die u kunt gebruiken om problemen met uw configuratie op te lossen.

Om een oplossing te vinden kunt u het logbestand controleren dat u in het Python-script maakt, met behulp van een lokale beheerder.

navigeren naar **Toepassingen**, klik op **Manager** in de **iox_web** toepassing en selecteer vervolgens het **tabblad Logs** zoals in de afbeelding weergegeven.



Log name	Timestamp	Log Size	Download
watchDog.log	Wed Mar 13 20:39:51 2019	97	download
webserver.log	Wed Mar 13 20:41:33 2019	39	download
container_log_iox_web.log	Wed Mar 13 20:39:51 2019	1684	download