إنشاء حزمة Docker IOx ونشرها لبنية IR1101 ARM

المحتويات

المقدمة

المتطلبات الأساسية

المتطلبات

المكونات المستخدمة

معلومات أساسية

التكوين

الباب الأول. إنشاء حزمة IOx لـ IR1101

1. قم بتثبيت عميل IOx واعداده على مضيف Linux

2. قم بتكوين وتهيئة Docker على جهاز كمبيوتر

3. قم بتثبيت حزم محاكاة مستخدم QEMU

4. قم بإجراء اختبار عند تشغيل حاوية Aarch64/ARV64v8 على جهاز Linux x86

5. اعداد الملفات لإنشاء حاوية داخل وبي Docker

6. إنشاء حاوية Docker

7. إنشاء حزمة IOx

الباب الثاني. تكوين IR1101 لـ IOx

1. تمكين واجهة الويب لـ IOx و Local Manager

2. تكوين شبكات IOx

الباب الثالث. Access Local Manager ونشر تطبيق IOx

التحقق من الصحة

استكشاف الأخطاء وإصلاحها

المقدمة

يوضح هذا المستند كيفية إعداد حزمة IOx قائمة على Docker وتكوينها ونشرها لبوابة إنترنت الأشياء (IOt) المستندة إلى IR1101 ARM.

المتطلبات الأساسية

المتطلبات

توصي Cisco بأن تكون لديك معرفة بالمواضيع التالية:

- لينكس
- الحاويات
- IOx

المكونات المستخدمة

تستند المعلومات الواردة في هذا المستند إلى إصدارات البرامج والمكونات المادية التالية:

- IR1101 القابل للوصول إليه عبر القشرة الآمنة (SSH)
  - تم تكوين عنوان IP
  - الوصول إلى الجهاز مع امتياز 15 مستعمل
- مضيف Linux (يتم إستخدام تثبيت الحد الأدنى ل Debian 9 (ممتد) لهذه المقالة)
- ملفات تثبيت عميل IOx التي يمكن تنزيلها من: https://software.cisco.com/download/release.html?mdfid=286306005&softwareid=286306762

تم إنشاء المعلومات الواردة في هذا المستند من الأجهزة الموجودة في بيئة معملية خاصة. بدأت جميع الأجهزة المُستخدمة في هذا المستند بتكوين ممسوح (افتراضي). إذا كانت شبكتك مباشرة، فتأكد من فهمك للتأثير المحتمل لأي أمر.

معلومات أساسية

يختلف الطراز IR1101 قليلا مقارنة بمعظم الأنظمة الأساسية الأخرى للإدخال/الإخراج (IOx) حيث تستند هذه الأنظمة أساسا إلى الطراز x86. إن IR1101 يقوم على بنية ARM64v8 لذلك لا يمكنك نشر حاويات أو حزم IOx مبنية ل x86 على النظام الأساسي مباشرة. يبدأ هذا المستند من الصفر ويجهز بيئة بناء حاويات إرساء قائمة على تقنية ARM64v8، كما يشرح كيفية إنشاء هذه الحاويات وتعبئتها ونشرها على الطراز IR1101 باستخدام جهاز كمبيوتر طراز x86.

كمثال على ذلك، يتم إستخدام برنامج نصي صغير جدا من Python عبارة عن خادم ويب بسيط ويتم بناء حاوية Docker حولها لحزمها في النهاية لتشغيلها على IR1101. الشيء الوحيد الذي يقوم به خادم الويب هو الاستماع على منفذ محدد مسبقا (9000) وإرجاع صفحة بسيطة عندما يستلم طلب GET. وهذا يتيح لك إختبار إمكانية تشغيل الرمز الخاص بك، كما يسمح باختبار وصول الشبكة إلى تطبيق IOx بمجرد بدء تشغيله.

تم إنشاء الحزمة بواسطة أدوات Docker، مع إستخدام Alpine Linux. إن Alpine Linux هو صورة صغيرة من Linux (حوالي 5 ميغابايت)، والتي غالبا ما تستخدم كقاعدة لحاويات المرساة.

بما أن معظم الأجهزة المكتبية والمحمولة/الأجهزة الافتراضية (VM) الموجودة على الكمبيوتر المكتبي/المحمول تستند جميعها إلى الطراز x86، فإنك بحاجة إلى محاكاة بنية ARM64v8 على الجهاز القائم على الطراز x86 حيث يتم إنشاء الحاوية. يمكنك القيام بذلك بسهولة باستخدام محاكاة مستخدم Quick Emulator (QEMU). وهذا يسمح بتنفيذ الجوالات في بنية غير أصلية تماما كما يمكن تشغيلها على بنيتها الأصلية.

التكوين

الباب الاول. إنشاء حزمة IOx ل IR1101

1. قم بتثبيت عميل IOx وإعداده على مضيف Linux

تحتاج إلى iOxclient لحزم حاوية Docker كحزمة IOx بمجرد بنائها، لذا، فلنعد هذا أولا.

قم بنسخ حزمة IOXCLIENT أو تنزيلها. وهو متاح على الموقع التالي:
https://software.cisco.com/download/release.html?mdfid=286306005&softwareid=286306762.

```
jedepuyd@deb9:~$ scp jedepuyd@192.168.56.101:/home/jedepuyd/ioxclient_1.7.0.0_linux_amd64.tar.gz .
jedepuyd@192.168.56.101's password:
ioxclient_1.7.0.0_linux_amd64.tar.gz                          100% 4798KB  75.2MB/s   00:00
```

إستخراج الحزمة:

```
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
```

قم بإضافة المسار إلى متغير المسار حتى يكون متوفرا بدون إستخدام الموقع الكامل. إذا قمت بإعادة تشغيل الجهاز أو تبديل المستخدمين، فلا تنس تكرار هذه الخطوة:

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

إطلاق ioxclient للمرة الأولى من أجل إنشاء توصيف إجباري. نظرا لأنك تستخدم IOXCLIENT لأول مرة، يمكنك ترك القيم كقيمة افتراضية فقط لحزم حاوية Docker:

```
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. قم بتركيب وتجهيز بيئة Docker على جهاز بناء لينوكس

يستخدم هذا المستند لبناء حاوية من صورة قاعدة جبال الألب وتضمين الملفات الضرورية لحالة الاستخدام. تستند الخطوات المحددة إلى أدلة التثبيت الرسمية من إصدار مجتمع الإرساء (CE) دبيان: https://docs.docker.com/install/linux/docker-ce/debian/

تحديث قوائم الحزم على جهازك:

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

قم بتثبيت التبعيات لاستخدام repo الخاص ب Docker:

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

إضافة مفتاح GNU لخصوصية Docker (GPG) كمفتاح GPG صالح:

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

تحقق من بصمة الإصبع الخاصة بمفتاح GPG المثبت:

```
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]
```

إضافة طلب إعادة الشراء الثابت للمنسقين:

```
jedepuyd@deb9:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/debian $(l
```

قم بتحديث قوائم الحزم مرة أخرى وأنت موقوت بإضافة رسالة إعادة توجيه Docker:

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

تثبيت Docker:

```
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) ...
```

لكي تتمكن من الوصول إلى/تشغيل Docker كمستخدم عادي، أضف هذا المستخدم إلى مجموعة Docker وقم بتحديث عضوية المجموعة:

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

3. تثبيت حزم محاكاة مستخدم QEMU

بعد تثبيت Docker، تحتاج إلى تثبيت محاكي مستخدم QEMU. أستخدم محاكي QEMU ل ARM64v8 المرتبط بشكل ثابت من داخل حاوية Docker حتى تتمكن من تشغيل الحاوية المستهدفة مصممة لبنية ARM64v8. على جهاز Linux الذي يعمل بنظام x86، رغم أن الحاوية

قم بتركيب الحزم:

```
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) ...
```

بعد التثبيت، إليك محاكي QEMU المرتبطة بشكل ثابت والمتوفرة في /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
...
```

أولا اسم في القائمة، وه الاسم الذي تحتاجه: Aarch64 وه الاسم الرئيسي ل ARM64v8 ماظنل التشغيل Linux.

4. مق بإجارء إختبار دنع تشغيل حاوية Aarch64/ARV64v8 على زاهج Linux x86

الآن دعب أن أصبح كيدل Docker ومثبتات ثنائيات QEMU الضرورية، يمكنك إختبار ام إذا تنك قادرا على تشغيل حاوية Docker ممصمة ل ARM64v8 على الجهاز x86:

```
jedepuyd@deb9:~$ docker run -v /usr/bin/qemu-aarch64-static:/usr/bin/qemu-aarch64-static --rm -ti arm64
Unable to find image 'arm64v8/alpine:3.7' locally
3.7: Pulling from arm64v8/alpine
40223db5366f: Pull complete
Digest: sha256:a50c0cd3b41129046184591963a7a76822777736258e5ade8445b07c88bfdcc3
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
```

كما ترون في المخرجات، يتم الحصول على حاوية Arm64v8 نم جبال الألب وتصنيعها لتعمل مع إمكانية الوصول إلى المحاكي.

إذا قمت بطلب بنية الحاوية، يمكنك أن ترى أن التعليمات البرمجية قد تم تحويلها برمجيا ل aarch64. تماما كما يجب أن يكون القوس المستهدف للحاوية ل IR1101.

5. إعداد الملفات لإنشاء حاوية خادم ويب Docker

الآن دعب الانتهاء نم جميع التحضيرات، يمكنك إنشاء الملفات الضرورية لحاوية خادم الويب التي يجب تشغيلها على IR1101.

الملف الأول وه webserver.py، ووه البرنامج النصي Python الذي تريد تشغيله في الحاوية. امب أن هذا مجرد مثال، نم الواضح، أنك ستستبدل هذا مع الرمز الفعلي in order to ركضت في تطبيق IOx:

```
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()
```

يحتوي هذا الرمز على المنطق للكتابة إلى ملف سجل، متوفر للاستشارة من Local Manager.

الملف الثاني المطلوب هو Dockerfile. يحدد هذا كيفية إنشاء الحاوية:

```
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
```

يحدد Dockerfile كيفية إنشاء الحاوية. ابدأ من الصورة الأساسية Apline لـ ARM64v8، وانسخ الملحاكي في الحاوية، وشغل نقطة الوصول لإضافة حزمة Python ونسخ النص التنفيذي لخادم الويب إلى الحاوية.

الإعداد الأخير المطلوب قبل إنشاء الحاوية هو نسخ Qemu-aarch64-static إلى الدليل من حيث تقوم بإنشاء الحاوية:

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

6. إنشاء حاوية Docker

الآن بعد أن تم كل الإعداد، يمكنك بناء الحاوية باستخدام 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
```

كإختبار، قم بتشغيل الحاوية التي قمت بإنشائها للتو وتحقق مما إذا كان البرنامج النصي يعمل:

```
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 -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      13/qemu-aarch64-sta
/ # exit
```

كما ترى في هذا الإخراج، فإن بنية الحاوية هي Aarch64 المطلوبة. وبعد أن تقوم بتشغيل النص التنفيذي، سترى أنه يستمع للطلبات على المنفذ 9000.

7. إنشاء حزمة IOx

الحاوية جاهزة ليتم حزمها. قبل أن تطلب من عميل iXP القيام بذلك، تحتاج أولا إلى إنشاء واصف الحزمة: package.yaml.

يوضح هذا الملف كيفية ظهور الحزمة وعدد الموارد التي تحتاج لتشغيلها وما يجب تشغيله.

```
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: c1.tiny
    network:
      -
        interface-name: eth0
        ports:
          tcp: ["9000"]

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

وكما ترى، تم تعيين بنية وحدة المعالجة المركزية (CPU) على Aarch64. للحصول على الوصول إلى منفذ TCP 9000، أستخدم rootfs.tar كطرز وفي البدء، يمكنك تشغيل python/webserver.py.

آخر شيء يجب عمله قبل أن تتمكن من الحزم هو إستخراج rootfs.tar من حاوية Docker:

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

عند هذه النقطة، يمكنك إستخدام ioxclient لإنشاء حزمة IOx ل IR1101:

```
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 defini
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 : bdf5596a0747eae51bb0a1d2870fd09a5a16a098
Path:  package.yaml
SHA1 : e65a6fcbe96725dd5a09b60036448106acc0c138
Generated package manifest at  package.mf
Generating IOx Package..
Package generated at /home/jedepuyd/iox_aarch64_webserver/package.tar
```

في الوقت الحالي، توجد حزمة جاهزة للنشر على الطراز IR1101 كحزمة.tar. يشرح الجزء التالي كيفية إعداد الجهاز للنشر.

الباب الثاني. تكوين IR1101 ل IOx

1. تمكين واجهة الويب و IOx و Local Manager

الإدارة المحلية هي واجهة مستخدم رسومية (GUI) من أجل نشر تطبيقات IOx وتنشيطها وتشغيلها وإدارتها واستكشاف أخطائها وإصلاحها. بالنسبة ل IR1101، يتم تضمين في و واجهة الويب الخاصة بالإدارة العادية. لذا، عليك أن تقوم بتمكين ذلك أولا.

قم بإجراء هذه الخطوات على IR1101 لتمكين IOx وواجهة الويب.

```
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
```

السطر الأخير يضيف مستخدما لديه أذونات المستوى 15. لهذا المستخدم حق الوصول إلى واجهة الويب والمدير المحلي ل IOx.

2. تكوين شبكات IOx

قبل الوصول إلى واجهة الويب، دعنا نضيف التكوين المطلوب لشبكات IOx. يمكن العثور على معلومات أساسية في وثائق IR1101 الخاصة ب IOx:
https://www.cisco.com/c/en/us/td/docs/routers/access/1101/software/configuration/guide/b-cisco-ir1101-scg.html

وباختصار، يمكن لتطبيقات IOx الاتصال بالعالم الخارجي باستخدام واجهة VirtualPortGroup0 (المماثلة مع Gi2 على واجهات IR809 و Gi5).

```
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
```

بينما تقوم بتكوين واجهة VirtualPortGroup0 كترجمة عنوان الشبكة (NAT) داخل الجهاز، تطبيقات IOx للسماح بالاتصال بالواجهة الخارجي على Gi 0/0/0 يلزمك إضافة بيان IP nat ومنها باستخدام NAT:

```
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
```

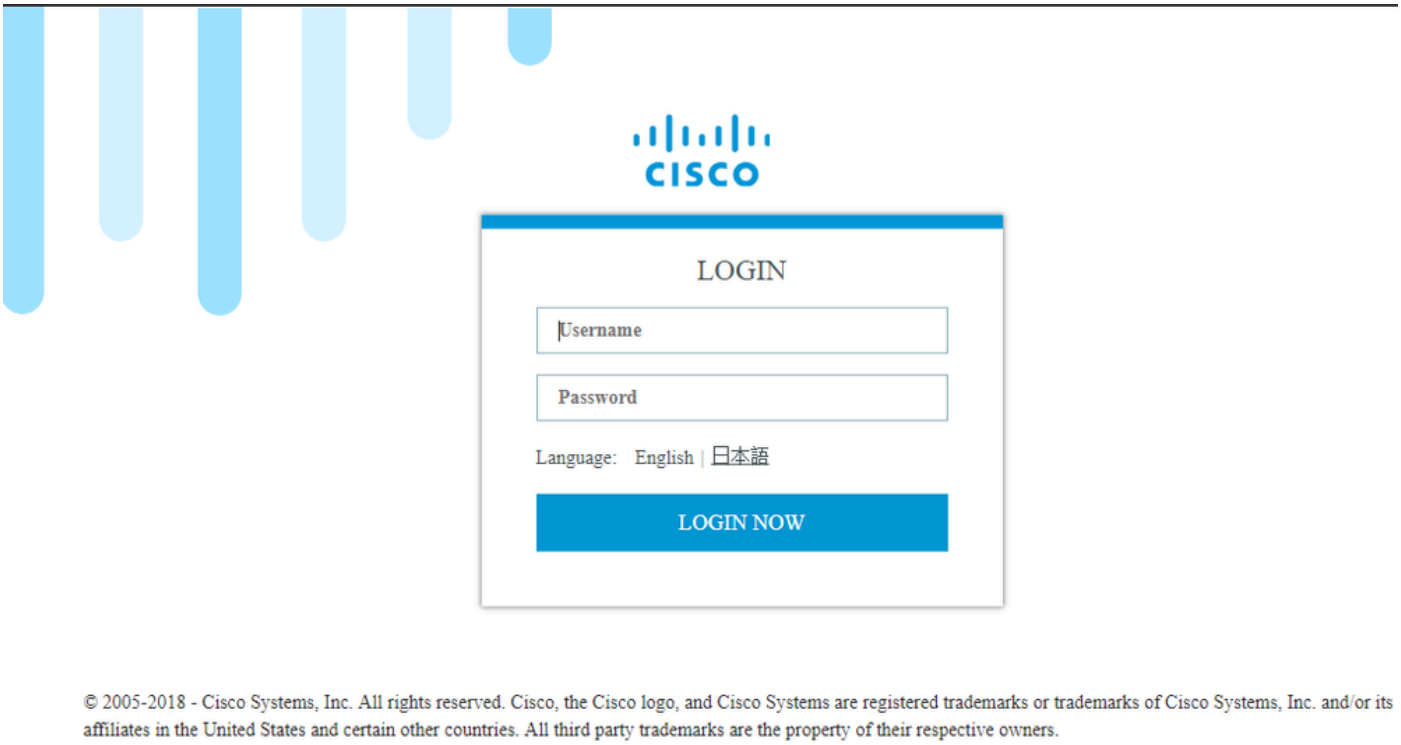
in order to سمحت منفذ إلى ميناء 9000 للحاوية، أي أنت يستطيع أعطته 192.168.1.15، أنت تحتاج أن يضيف ميناء للأمام:

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

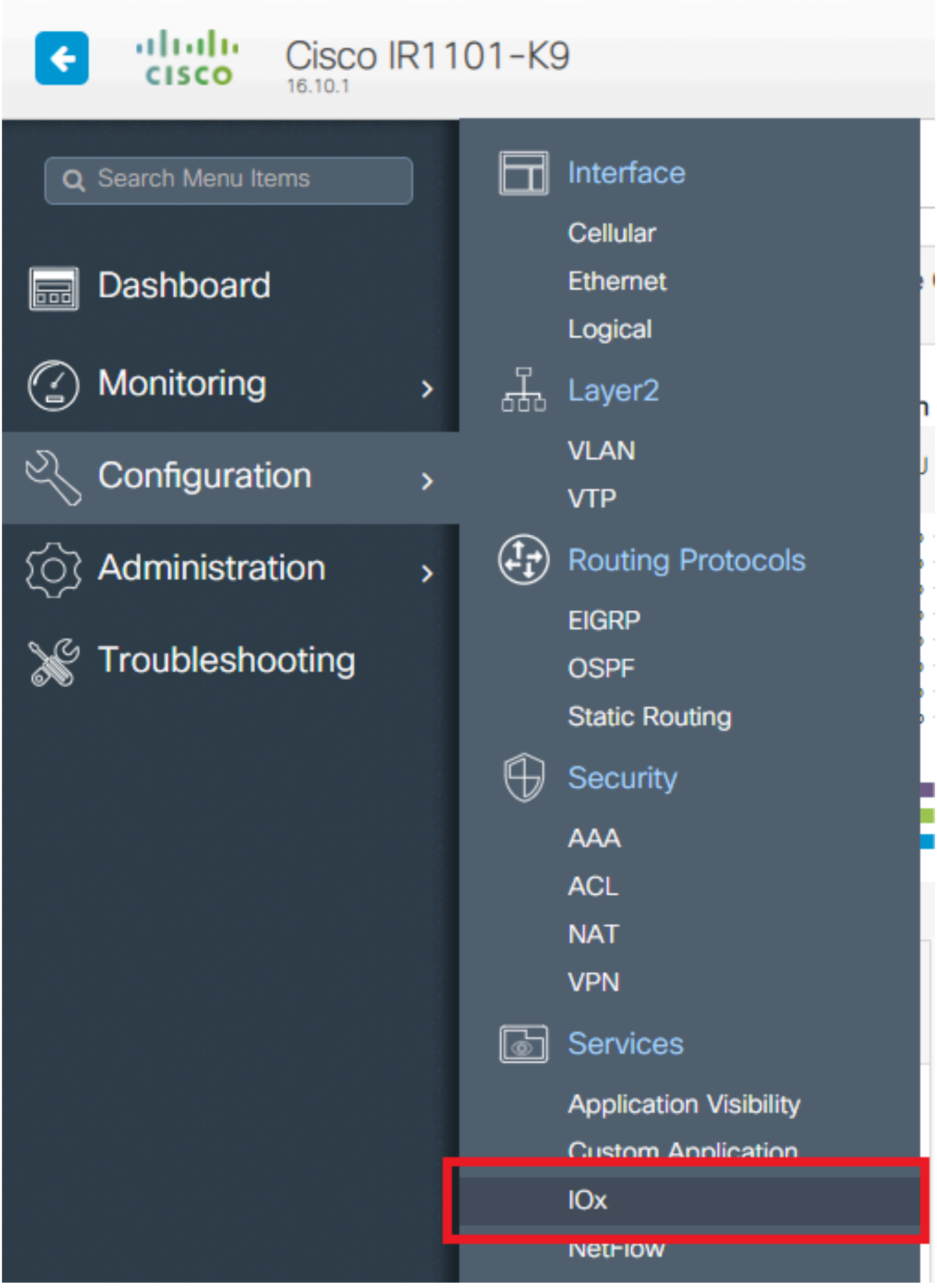
بالنسبة لهذا الدليل، أستخدم عناوين IP التي تم تكوينها بشكل ثابت لكل تطبيق IOx. إذا كنت ترغب في تعيين عناوين IP بشكل ديناميكي إلى التطبيقات، فأنت تحتاج إلى إضافة التكوين لخادم DHCP في الشبكة الفرعية ل VirtualPortGroup0.

الباب الثالث. Access Local Manager ونشر تطبيق IOx

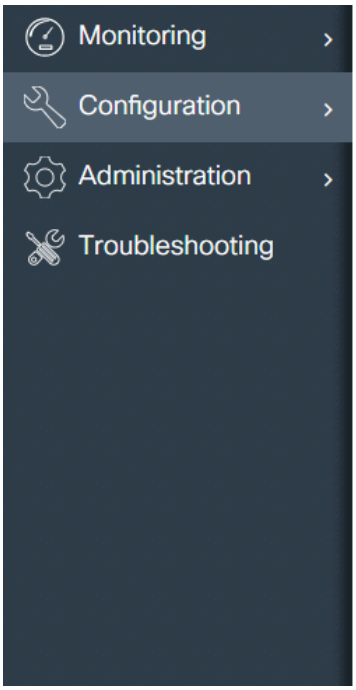
بعد إضافة هذه الخطوط إلى التكوين، يمكنك الوصول إلى IR1101 باستخدام واجهة الويب. انتقل إلى عنوان IP Gi 0/0/0 باستخدام المستعرض الخاص بك كما هو موضح في الصورة.

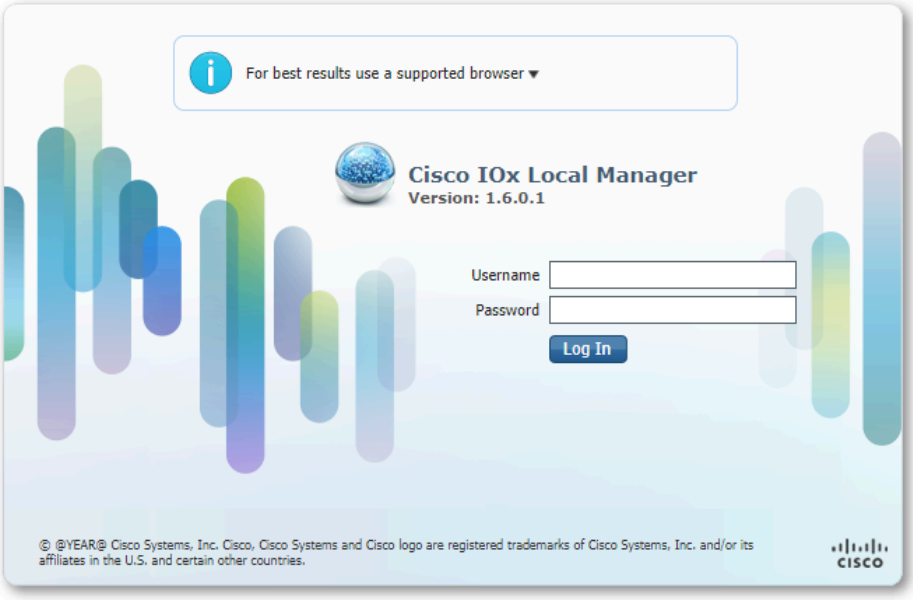


1. لتسجيل الدخول إلى واجهة الويب أستخدم حساب الامتياز 15 الذي تم إنشاؤه في الخطوة 1. والتصفح إلى Configuration - IOx كما هو موضح في الصورة.

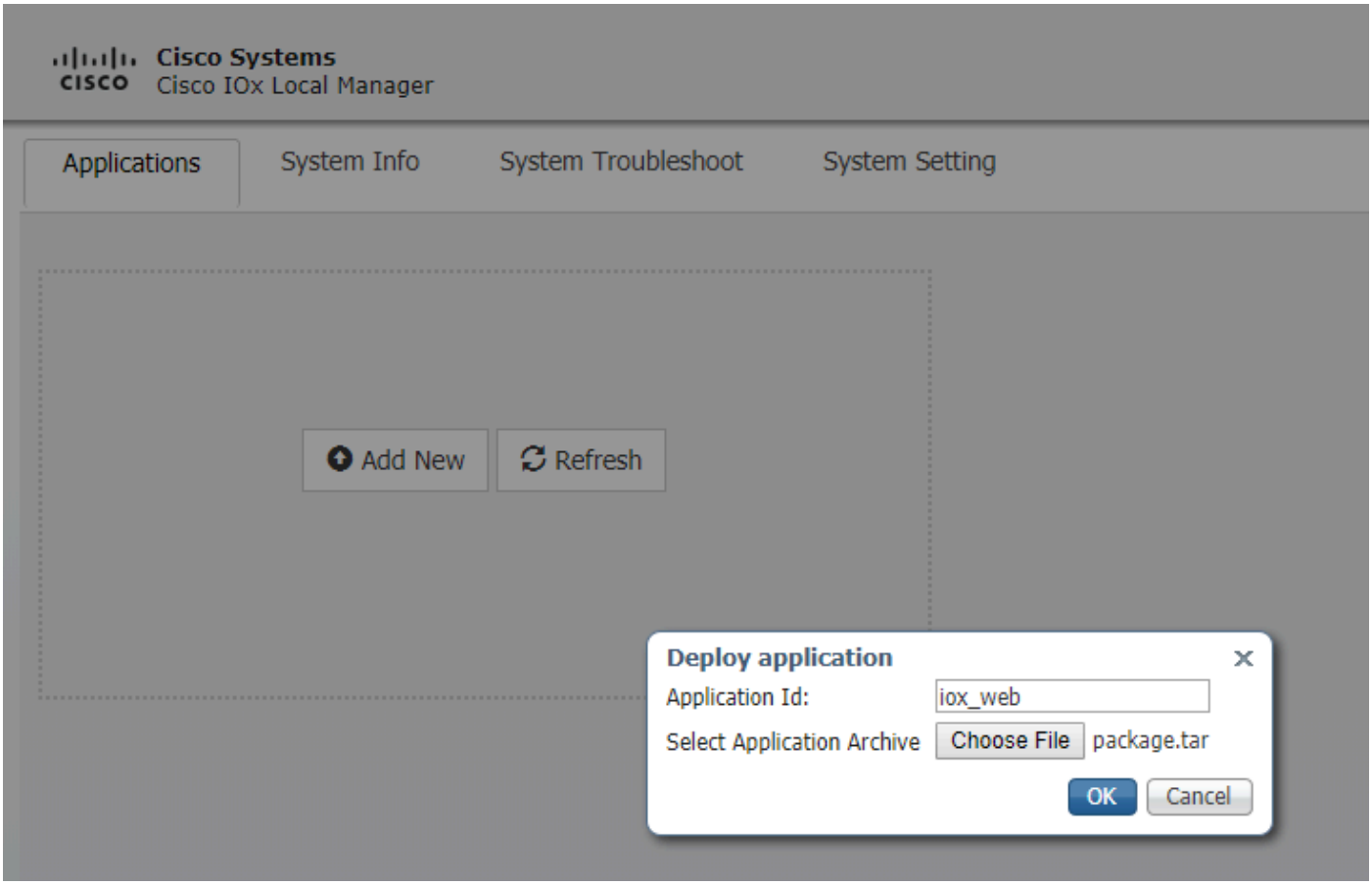

Cisco IR1101-K9
16.10.1
Search Menu Items
Dashboard
Monitoring
Configuration
Administration
Troubleshooting
Interface
Cellular
Ethernet
Logical
Layer2
VLAN
VTP
Routing Protocols
EIGRP
OSPF
Static Routing
Security
AAA
ACL
NAT
VPN
Services
Application Visibility
Custom Application
IOx
NetFlow

في تسجيل دخول "الإدارة المحلية ل IOx"، أستخدم الحساب نفسه للمتابعة كما هو موضح في الصورة.

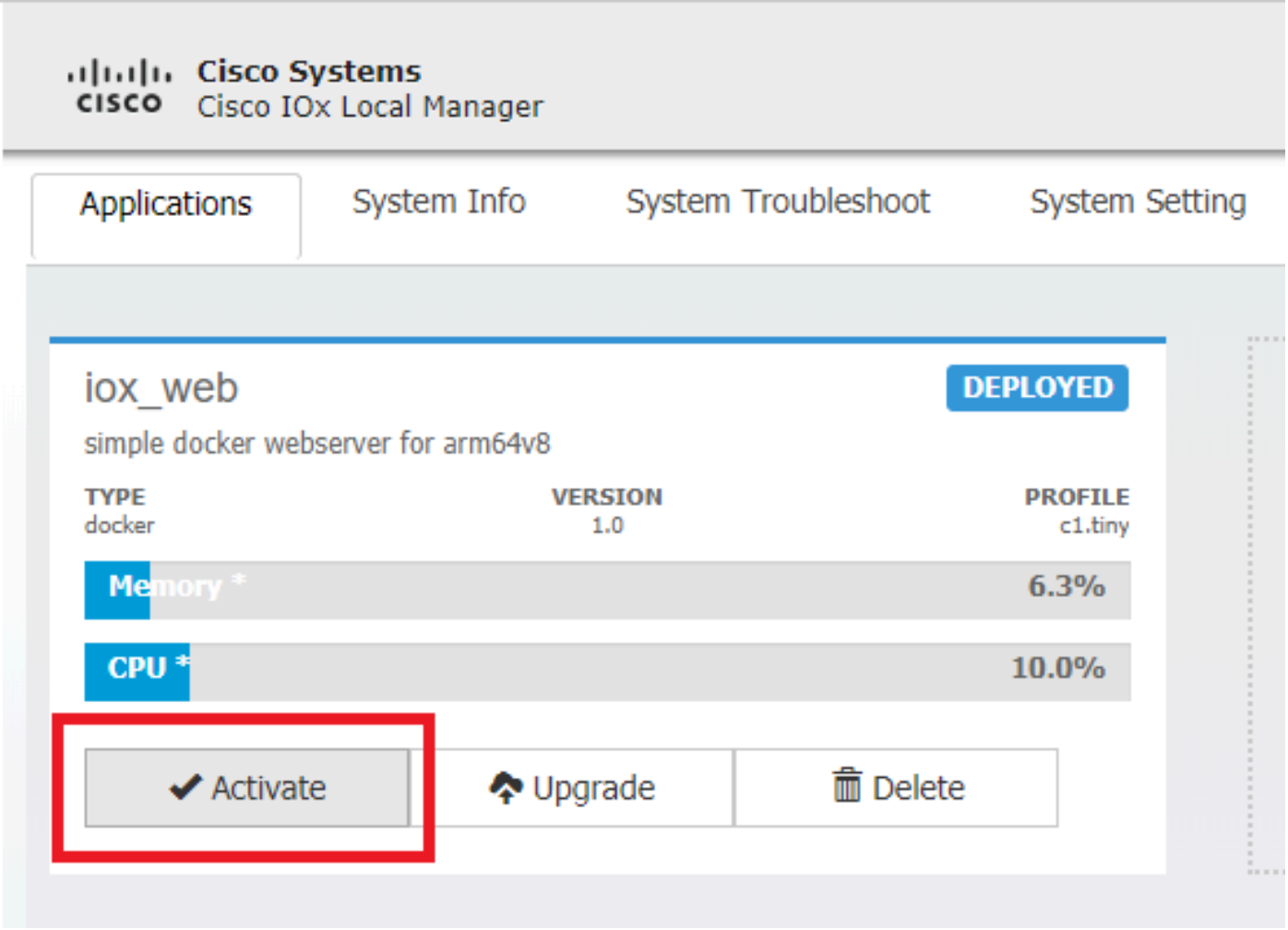




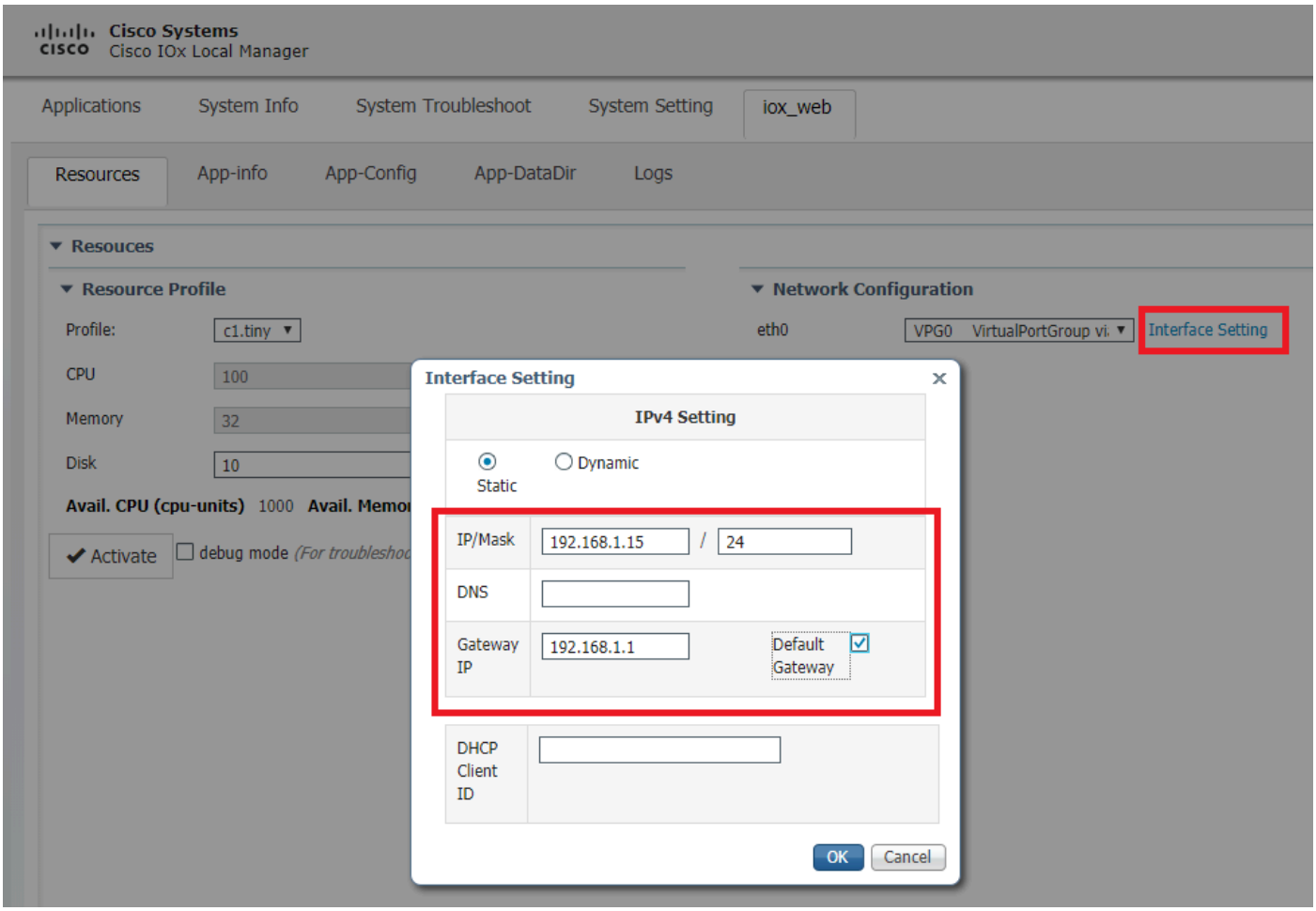
انقر فوق إضافة جديد، وحدد اسما لتطبيق IOx واختر package.tar الذي تم إنشاؤه في الجزء 1 كما هو موضح في الصورة.



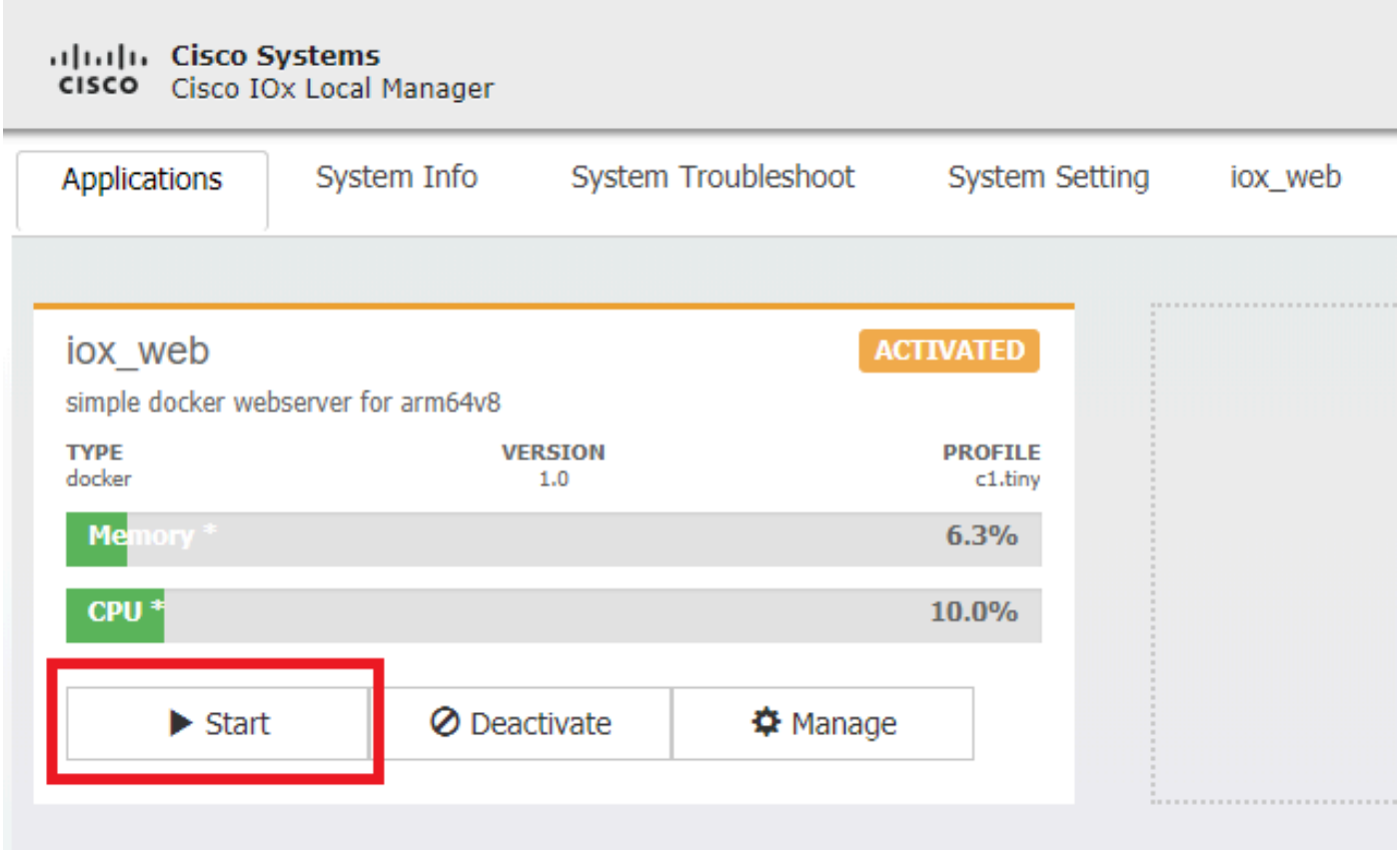
بمجرد تحميل الحزمة، يمكنك تنشيطها كما هو موضح في الصورة.



في علامة التبويب موارد، افتح إعداد الواجهة لتحديد IP الثابت الذي تريد تعيينه للتطبيق كما هو موضح في الصورة.



انقر فوق موافق، ثم تنشيط. بمجرد اكتمال العملية، قم بالانتقال للخلف إلى صفحة الإدارة المحلية الرئيسية (زر تطبيقات في القائمة العليا)، ثم قم بتشغيل التطبيق كما هو موضح في الصورة.



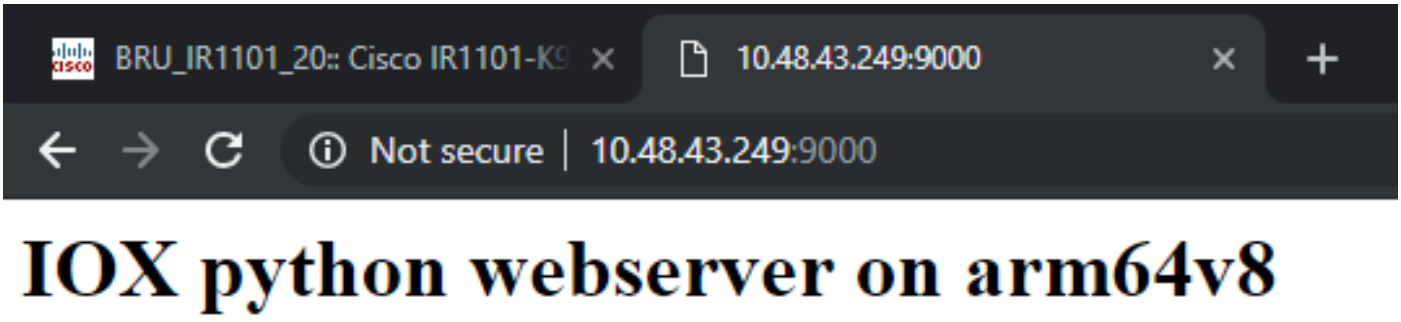
بعد المرور بهذه الخطوات، يتم تشغيل التطبيق الخاص بك ويتوفر عبر المنفذ 9000 باستخدام واجهة Gi 0/0/0 الخاصة ب IR1101.

التحقق من الصحة

استخدم هذا القسم لتأكيد عمل التكوين بشكل صحيح.

in order to دقق، أنت يستطيع نفذت العنوان من ال gi 0/0/0 قارن على ال ir1101 مع الإستعاملمن ميناء 9000.

إذا سار كل شيء على ما يرام، هذا يظهر، كما تم إنشاؤه في نص بايثون.

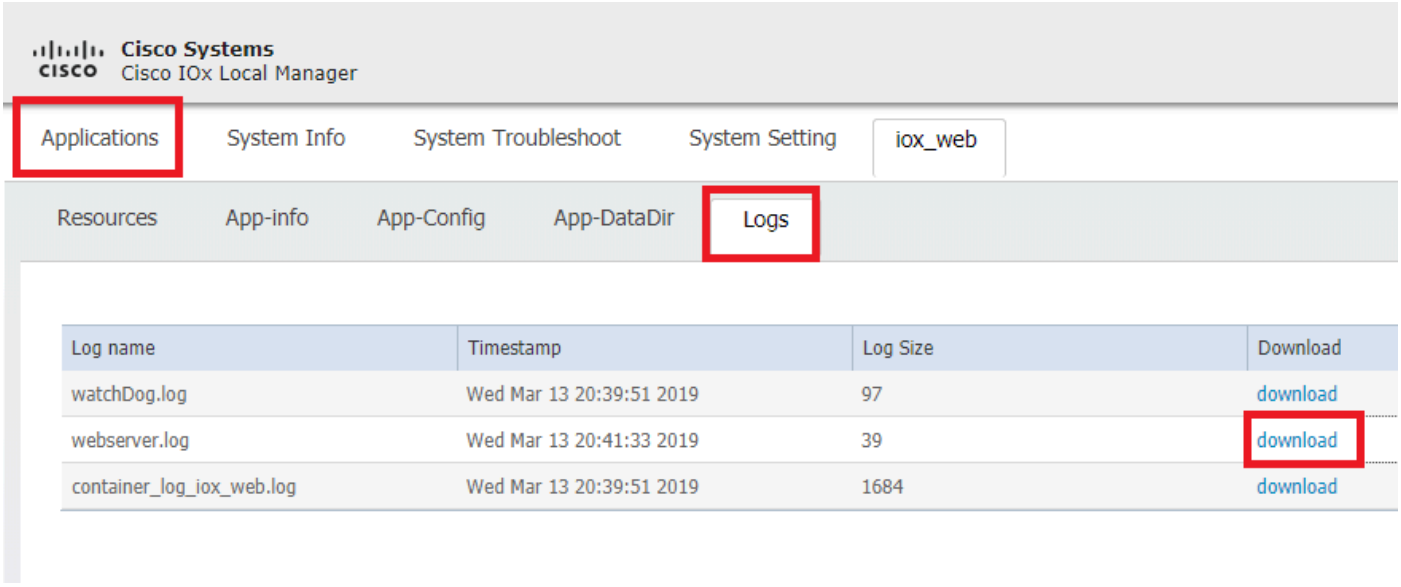


استكشاف الأخطاء وإصلاحها

يوفر هذا القسم معلومات يمكنك إستخدامها لاستكشاف أخطاء التكوين وإصلاحها.

لاستكشاف الأخطاء وإصلاحها، يمكنك التحقق من ملف السجل الذي تقوم بإنشائه في البرنامج النصي Python باستخدام مدير محلي.

انتقل إلى التطبيقات، انقر فوق إدارة في تطبيق iox_web، ثم حدد علامة التبويب السجلات كما هو موضح في الصورة.



حول هذه الترجمة
ترجمت Cisco هذا المستند باستخدام مجموعة من التقنيات الآلية والبشرية لتقديم محتوى دعم للمستخدمين في جميع أنحاء العالم بلغتهم الخاصة. يُرجى ملاحظة أن أفضل ترجمة آلية لن تكون دقيقة كما هو الحال مع الترجمة الاحترافية التي يقدمها مترجم محترف. تخلي Cisco Systems مسئوليتها عن دقة هذه الترجمات وتُوصي بالرجوع دائمًا إلى المستند الإنجليزي الأصلي (الرابط متوفر).