使用Vagrant和Virtualbox/VMWare構建IOx應用

目錄

簡介

必要條件

- Windows/ MAC Intel/ Linux
- 基於MAC ARM - M1/M2/M3

使用Vagrant建立環境的程式

- 行動摘要

建立自訂IOx應用程式的程式

部署IOx應用程式

疑難排解

簡介

本文檔介紹如何使用Vagrant和Virtualbox構建IOx應用程式，以及如何在IOx本地管理器GUI中部署它們。

必要條件

Windows/ MAC Intel/ Linux

- Git
- 流浪漢
- Virtualbox

基於MAC ARM - M1/M2/M3

- Git
- 流浪漢
- VMWare融合
- vagrant-vmware-desktop外掛

下載：

- 流浪漢
- VirtualBox

使用Vagrant建立環境的程式

行動摘要

- vagrantfile 配置基於其主機體系結構設定VM環境。
- 它根據體系結構將VM配置為使用VMware Fusion或VirtualBox
- 它為VM調配必要的軟體和工具，包括QEMU（快速EMUlator）、Docker和ioxclient。
- 配置自動為amd64目標思科平台裝置構建一個iperf應用示例。

步驟 1.在本地系統中克隆Github儲存庫：

```
git clone https://github.com/suryasundarraj/cisco-iox-app-build.git
```

或者，複製配置盤櫃的內容並貼上到「Vagrantfile」中。這樣會在本機系統中建立名為「Vagrantfile」的檔案：

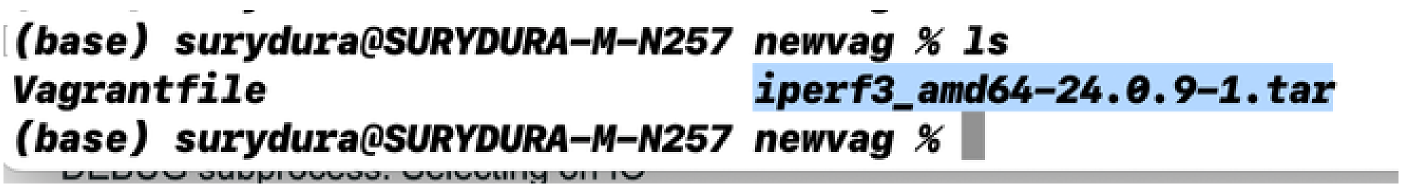
```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
# you're doing.
Vagrant.configure('2') do |config|
  arch = `arch`.strip()
  if arch == 'arm64'
    puts "This appears to be an ARM64 machine! ..."
    config.vm.box = 'gyptazy/ubuntu22.04-arm64'
    config.vm.boot_timeout = 600
    config.vm.provider "vmware_fusion" do |vf|
      #vf.gui = true
      vf.memory = "8192"
      vf.cpus = "4"
    end
    config.vm.define :ioxappbuild
  else
    puts "Assuming this to be an Intel x86 machine! ..."
    config.vm.box = "bento/ubuntu-22.04"
    config.vm.network "public_network", bridge: "ens192"
    config.vm.boot_timeout = 600
    config.vm.provider "virtualbox" do |vb|
      #vb.gui = true
      vb.memory = "8192"
      vb.cpus = "4"
    end
    config.vm.define :ioxappbuild
  end

  config.vm.provision "shell", inline: <<-SHELL
    #!/bin/bash
    # apt-cache madison docker-ce
    export VER="5:24.0.9-1~ubuntu.22.04~jammy"
    echo "!!! installing dependencies and packages !!!"
    apt-get update
    apt-get install -y ca-certificates curl unzip git pcregrep
    install -m 0755 -d /etc/apt/keyrings
    curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
    chmod a+r /etc/apt/keyrings/docker.asc
    echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://downlo
    apt-get update
    apt-get install -y qemu binfmt-support qemu-user-static
    apt-get install -y docker-ce=$VER docker-ce-cli=$VER docker-ce-rootless-extras=$VER containerd.io d
    # apt-get install -y docker.io docker-compose docker-buildx
    usermod -aG docker vagrant
    echo "!!! generating .ioxclientcfg.yaml file !!!"
    echo 'global:' > /home/vagrant/.ioxclientcfg.yaml
    echo '  version: "1.0"' >> /home/vagrant/.ioxclientcfg.yaml
    echo '  active: default' >> /home/vagrant/.ioxclientcfg.yaml
    echo '  debug: false' >> /home/vagrant/.ioxclientcfg.yaml
    echo '  fogportalprofile:' >> /home/vagrant/.ioxclientcfg.yaml
    echo '    fogpip: ""' >> /home/vagrant/.ioxclientcfg.yaml
    echo '    fogpport: ""' >> /home/vagrant/.ioxclientcfg.yaml
    echo '    fogpapiprefix: ""' >> /home/vagrant/.ioxclientcfg.yaml
    echo '    fogpurlscheme: ""' >> /home/vagrant/.ioxclientcfg.yaml
    echo '  dockerconfig:' >> /home/vagrant/.ioxclientcfg.yaml
    echo '    server_uri: unix:///var/run/docker.sock' >> /home/vagrant/.ioxclientcfg.yaml
    echo '    api_version: "1.22"' >> /home/vagrant/.ioxclientcfg.yaml
    echo 'author:' >> /home/vagrant/.ioxclientcfg.yaml
    echo '  name: |' >> /home/vagrant/.ioxclientcfg.yaml
    echo '    Home' >> /home/vagrant/.ioxclientcfg.yaml
    echo '  link: localhost' >> /home/vagrant/.ioxclientcfg.yaml
    echo 'profiles: {default: {host_ip: 127.0.0.1, host_port: 8443, auth_keys: cm9vdDpyb290,' >> /home/
    echo '    auth_token: "", local_repo: /software/downloads, api_prefix: /iox/api/v2/hosting/,' >> /h
    echo '    url_scheme: https, ssh_port: 2222, rsa_key: "", certificate: "", cpu_architecture: "",' >
    echo '    middleware: {mw_ip: "", mw_port: "", mw_baseuri: "", mw_urlscheme: "", mw_access_token: "
    echo '    conn_timeout: 1000, client_auth: "no", client_cert: "", client_key: ""}}' >> /home/vagran
    cp /home/vagrant/.ioxclientcfg.yaml /root/.ioxclientcfg.yaml
    chown vagrant:vagrant /home/vagrant/.ioxclientcfg.yaml
    arch=$(uname -m)
    if [[ $arch == x86_64 ]]; then
      # download page https://developer.cisco.com/docs/iox/iox-resource-downloads/
      echo "!!! downloading and extracting ioxclient for x86_64 architecture !!!"
      curl -O https://pubhub.devnetcloud.com/media/iox/docs/artifacts/ioxclient/ioxclient-v1.17.0.0/iox
      tar -xvf /home/vagrant/ioxclient_1.17.0.0_linux_amd64.tar.gz
      cp /home/vagrant/ioxclient_1.17.0.0_linux_amd64/ioxclient /usr/local/bin/ioxclient
      rm -rv /home/vagrant/ioxclient_1.17.0.0_linux_amd64
    elif  [[ $arch = aarch64 ]]; then
      # download page https://developer.cisco.com/docs/iox/iox-resource-downloads/
      echo "!!! downloading and extracting ioxclient for arm64 architecture !!!"
      curl -O https://pubhub.devnetcloud.com/media/iox/docs/artifacts/ioxclient/ioxclient-v1.17.0.0/iox
      tar -xvf /home/vagrant/ioxclient_1.17.0.0_linux_arm64.tar.gz
      cp /home/vagrant/ioxclient_1.17.0.0_linux_arm64/ioxclient /usr/local/bin/ioxclient
      rm -rv /home/vagrant/ioxclient_1.17.0.0_linux_arm64
    fi
    chown vagrant:vagrant /usr/local/bin/ioxclient
    echo "!!! pulling and packaging the app for x86_64 architecture !!!"
    docker pull --platform=linux/amd64 mlabbe/iperf3
    ioxclient docker package mlabbe/iperf3 .
    cp package.tar /vagrant/iperf3_amd64-$(echo $VER | pcregrep -o1 ':([0-9.-]+)~').tar
  SHELL
end
```

步驟 2.確保「export VER="5:24.0.9-1~ubuntu.22.04~jammy」行未增加註釋，並註釋所有其他導出語句。這對應於您想要在此Vagrant環境中安裝的Docker Engine版本：

```
cisco@cisco-virtual-machine:~/Desktop/ioxappbuild$ cat Vagrantfile | grep 'export' | grep -v '#'
export VER="5:24.0.9-1~ubuntu.22.04~jammy"
```

步驟 3.使用Vagrantfile所在目錄中的vagrant up命令啟動Vagrant環境，並觀察為amd64 tar檔案成功構建的iperf IOx應用程式：

```
vagrant up
```



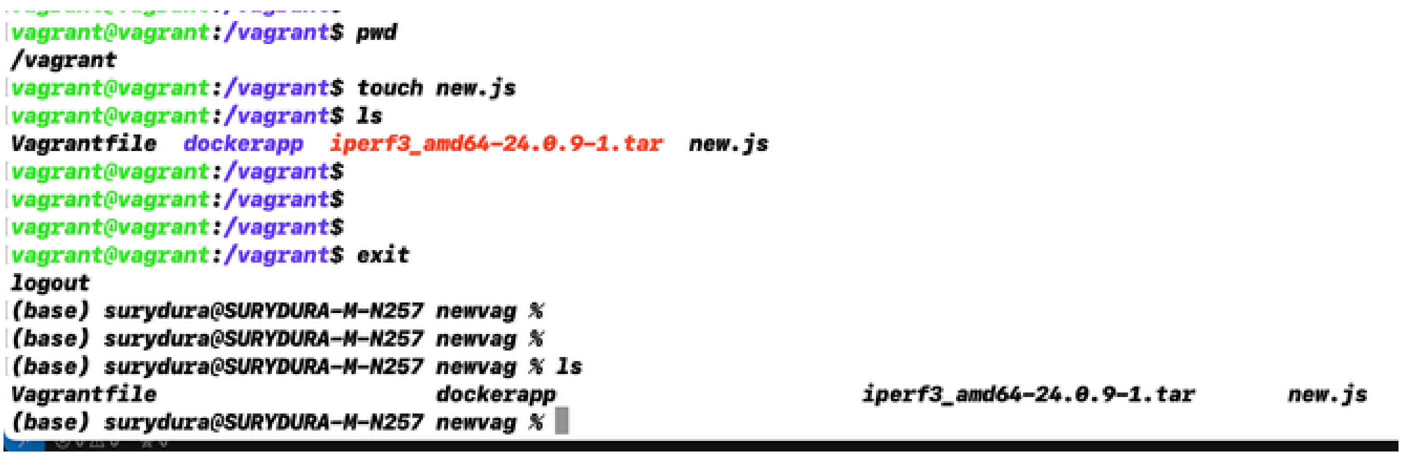
```
(base) surydura@SURYDURA-M-N257 newvag % ls
Vagrantfile                        iperf3_amd64-24.0.9-1.tar
(base) surydura@SURYDURA-M-N257 newvag %
```

建立自訂IOx應用程式的程式

本節介紹如何使用vagrant環境構建自定義IOx應用程式。

註：VM中的目錄「/vagrant」與主機系統中包含「Vagrantfile」的目錄同步。

如圖所示，new.js檔案在VM內建立，也可以在主機系統上訪問：



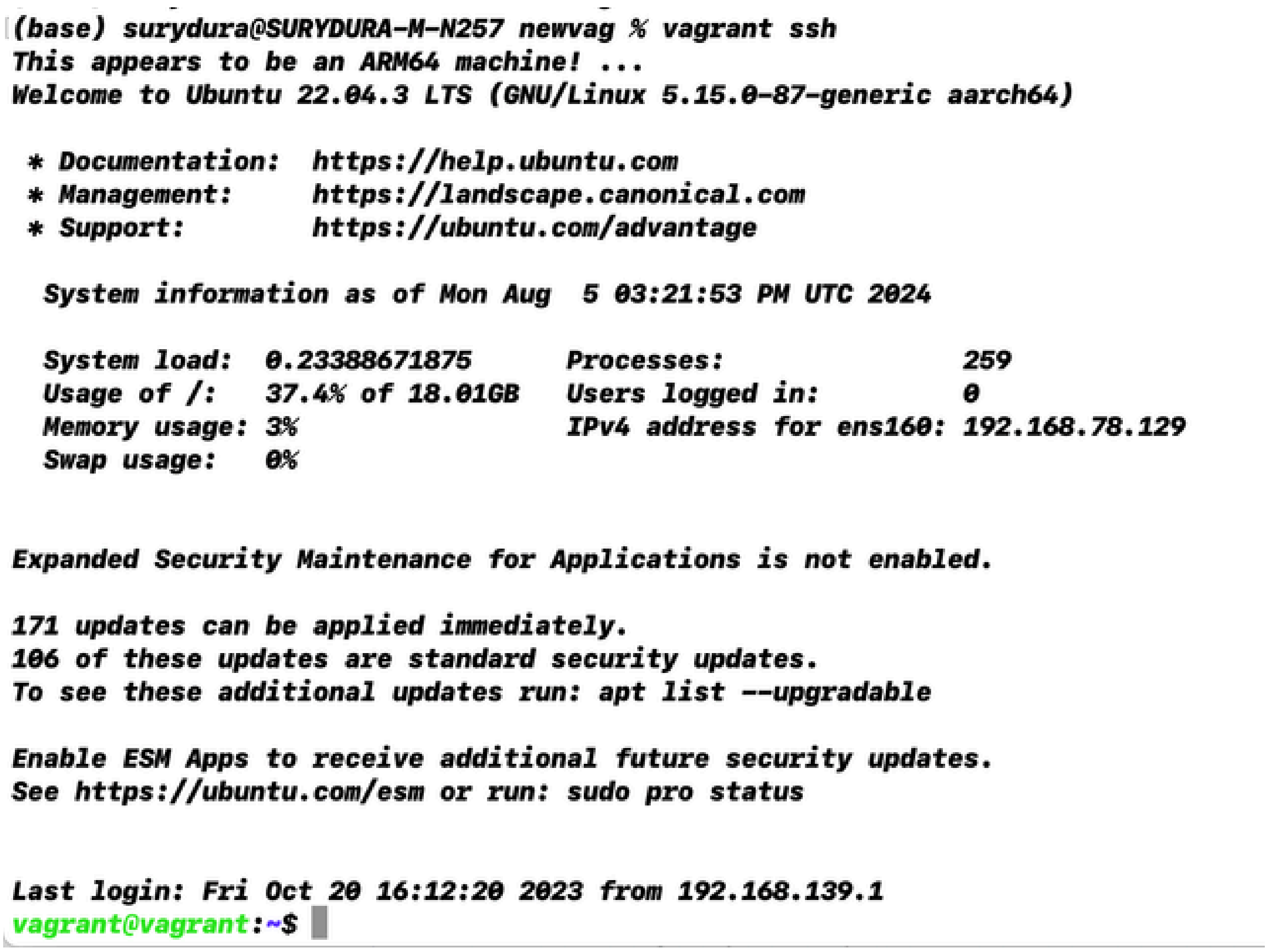
```
vagrant@vagrant:/vagrant$ pwd
/vagrant
vagrant@vagrant:/vagrant$ touch new.js
vagrant@vagrant:/vagrant$ ls
Vagrantfile  dockerapp  iperf3_amd64-24.0.9-1.tar  new.js
vagrant@vagrant:/vagrant$
vagrant@vagrant:/vagrant$
vagrant@vagrant:/vagrant$
vagrant@vagrant:/vagrant$ exit
logout
(base) surydura@SURYDURA-M-N257 newvag %
(base) surydura@SURYDURA-M-N257 newvag %
(base) surydura@SURYDURA-M-N257 newvag % ls
Vagrantfile                     dockerapp                    iperf3_amd64-24.0.9-1.tar      new.js
(base) surydura@SURYDURA-M-N257 newvag %
```

步驟 1.將範例應用程式複製到「Vagrantfile」所在的相同資料夾。本示例中使用的是iox-multiarch-nginx-nyancat-sample應用程式：

```
git clone https://github.com/etychon/iox-multiarch-nginx-nyancat-sample.git
```

步驟 2.SSH進入流浪者機器：

```
vagrant ssh
```

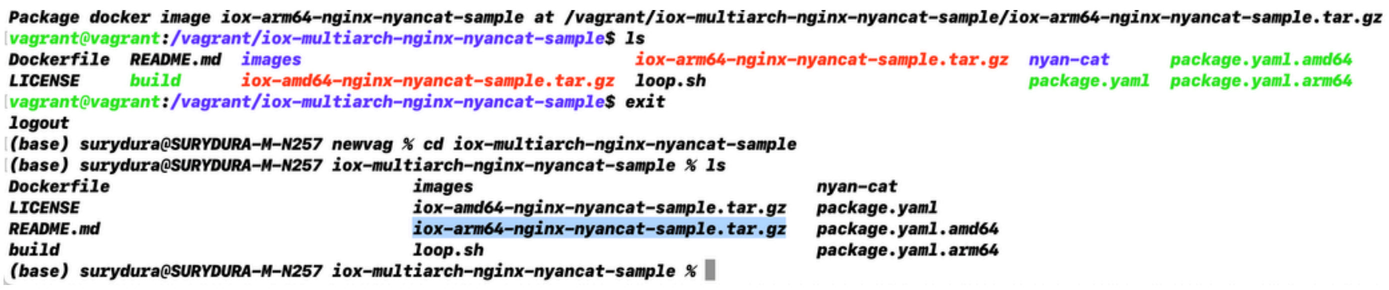


```
(base) surydura@SURYDURA-M-N257 newvag % vagrant ssh
This appears to be an ARM64 machine! ...
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-87-generic aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Mon Aug  5 03:21:53 PM UTC 2024

  System load:  0.23388671875      Processes:               259
  Usage of /:   37.4% of 18.01GB   Users logged in:         0
  Memory usage: 3%                 IPv4 address for ens160: 192.168.78.129
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

171 updates can be applied immediately.
106 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


Last login: Fri Oct 20 16:12:20 2023 from 192.168.139.1
vagrant@vagrant:~$
```

步驟 3.建置應用程式：

```
cd /vagrant/iox-multiarch-nginx-nyancat-sample/
chmod +x build
sh ./build
```

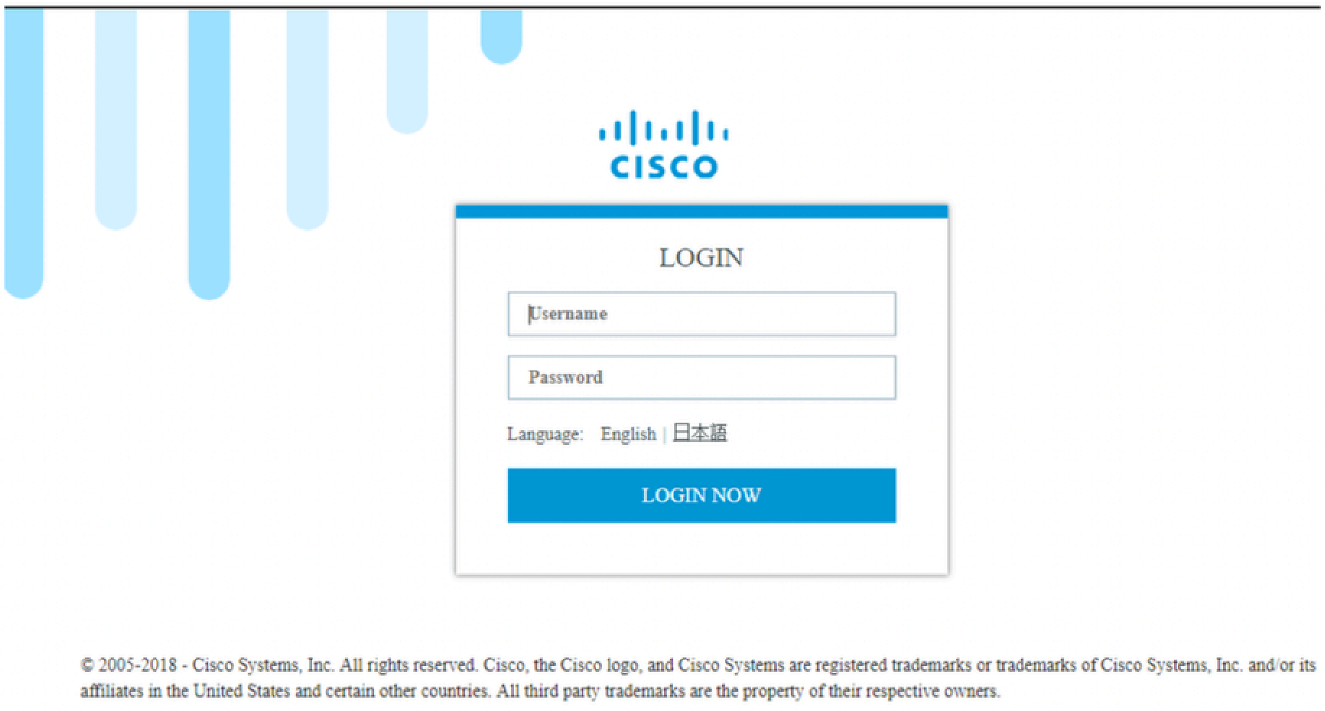
建置程式完成後，您現在擁有兩個可供部署的IOx應用程式(amd64為「iox-amd64-nginx-nyancat-sample.tar.gz」，目標平台為「iox-arm64-nginx-nyancat-sample.tar.gz」)：



```
Package docker image iox-arm64-nginx-nyancat-sample at /vagrant/iox-multiarch-nginx-nyancat-sample/iox-arm64-nginx-nyancat-sample.tar.gz
vagrant@vagrant:/vagrant/iox-multiarch-nginx-nyancat-sample$ ls
Dockerfile  README.md  images                                   iox-arm64-nginx-nyancat-sample.tar.gz  nyan-cat      package.yaml.amd64
LICENSE     build      iox-amd64-nginx-nyancat-sample.tar.gz    loop.sh                                package.yaml  package.yaml.arm64
vagrant@vagrant:/vagrant/iox-multiarch-nginx-nyancat-sample$ exit
logout
(base) surydura@SURYDURA-M-N257 newvag % cd iox-multiarch-nginx-nyancat-sample
(base) surydura@SURYDURA-M-N257 iox-multiarch-nginx-nyancat-sample % ls
Dockerfile                              images                                  nyan-cat
LICENSE                                 iox-amd64-nginx-nyancat-sample.tar.gz   package.yaml
README.md                               iox-arm64-nginx-nyancat-sample.tar.gz   package.yaml.amd64
build                                   loop.sh                                 package.yaml.arm64
(base) surydura@SURYDURA-M-N257 iox-multiarch-nginx-nyancat-sample %
```

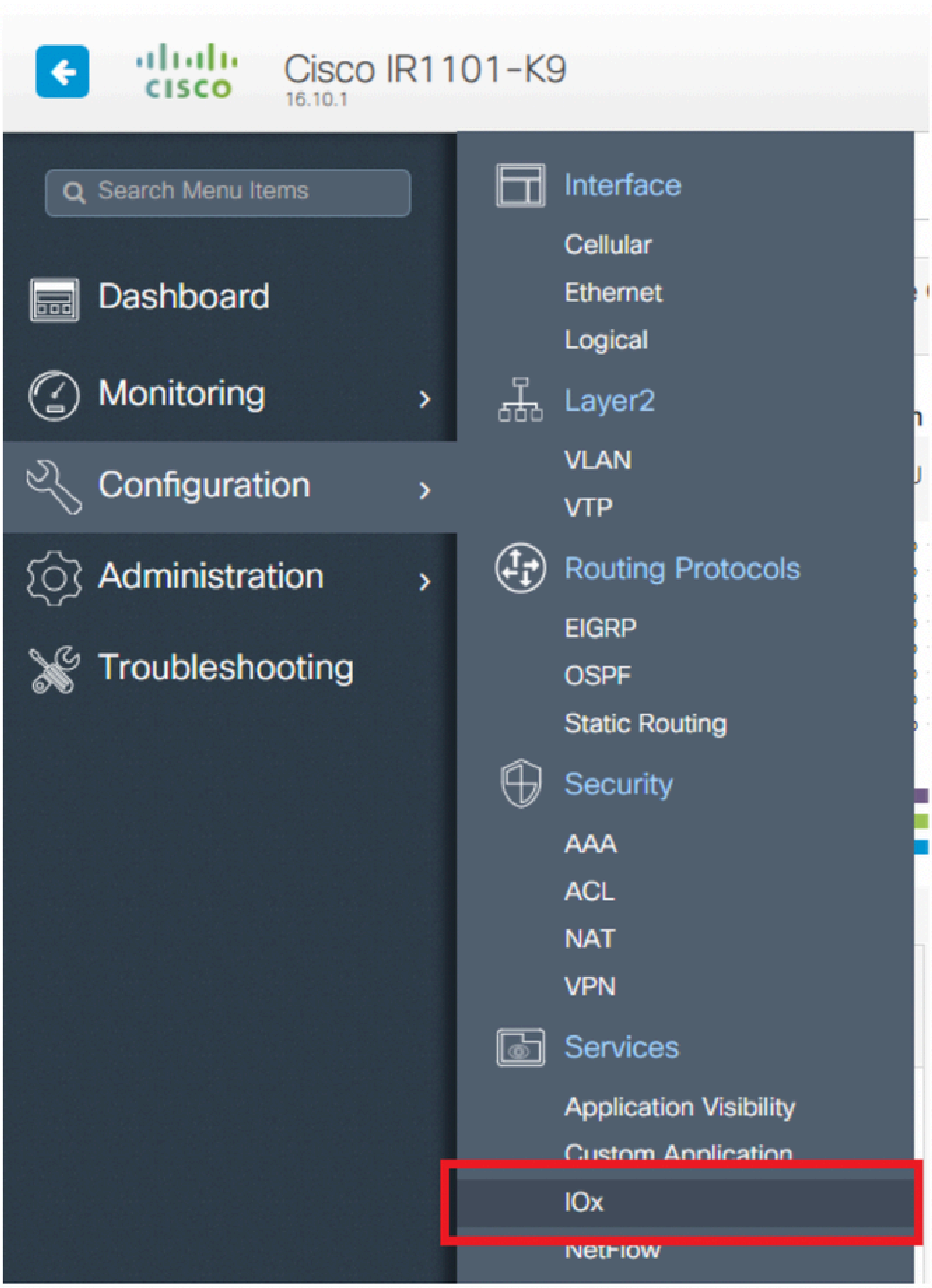
部署IOx應用程式

步驟 1.使用Web介面訪問IR1101：

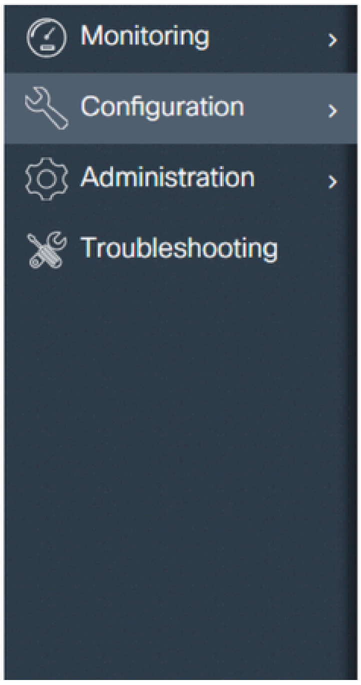


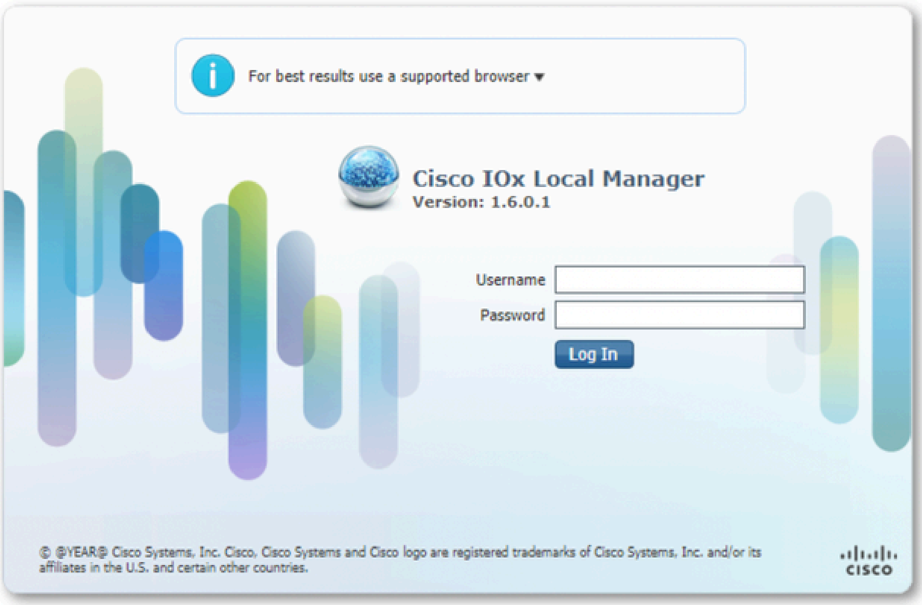
© 2005-2018 - Cisco Systems, Inc. All rights reserved. Cisco, the Cisco logo, and Cisco Systems are registered trademarks or trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries. All third party trademarks are the property of their respective owners.

步驟 2.使用許可權15帳戶：

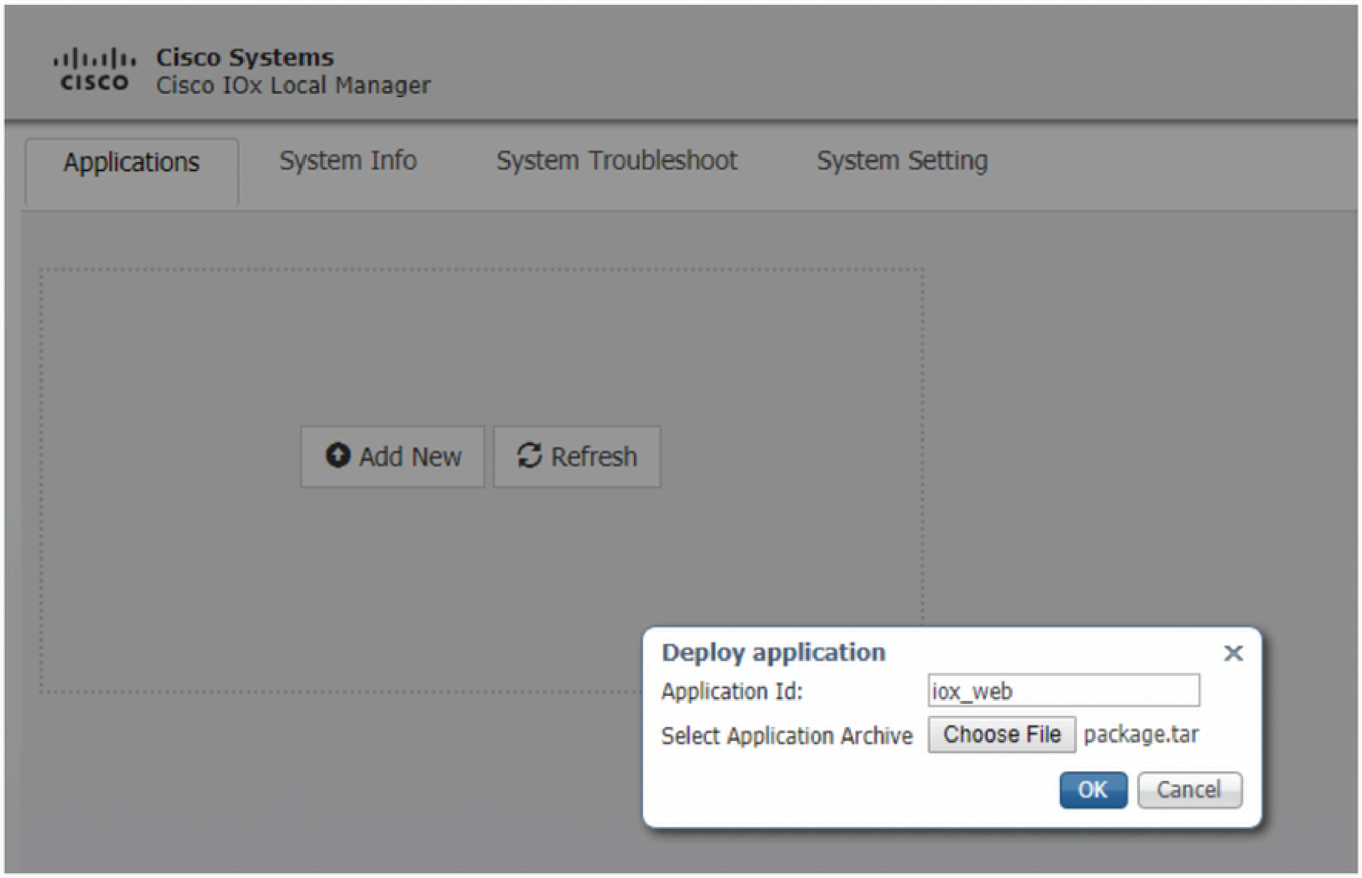


步驟 3.在IOx Local Manager登入中，使用相同的帳戶繼續，如圖所示：



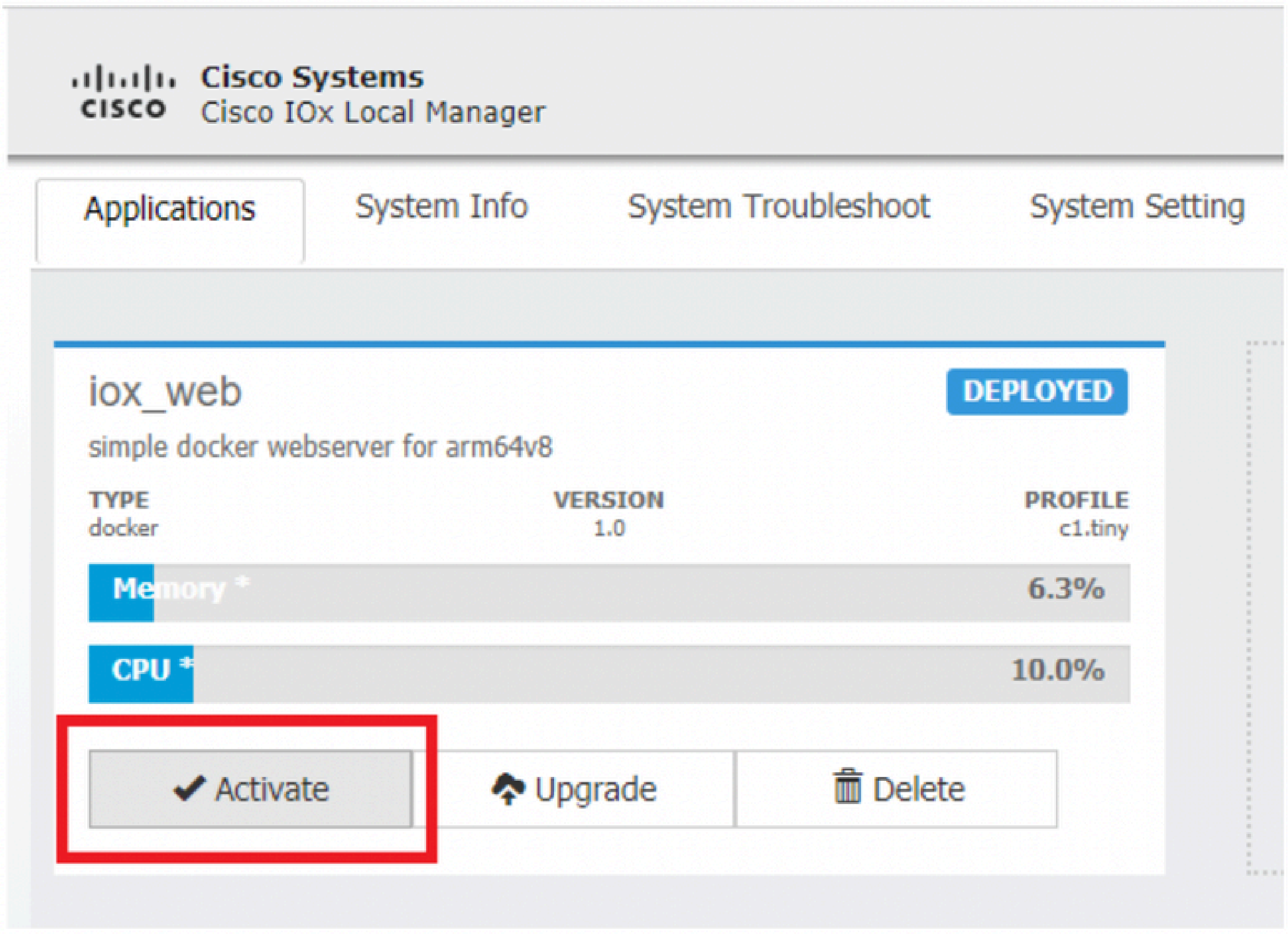

For best results use a supported browser
Cisco IOx Local Manager
Version: 1.6.0.1
Username
Password
Log In
© @YEAR@ Cisco Systems, Inc. Cisco, Cisco Systems and Cisco logo are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the U.S. and certain other countries.

步驟 4.按一下Add New，選擇IOx應用程式的名稱，並選擇Procedure to Set Up Environment Using Vagrant部分的第3步中構建的package.tar，如下圖所示：

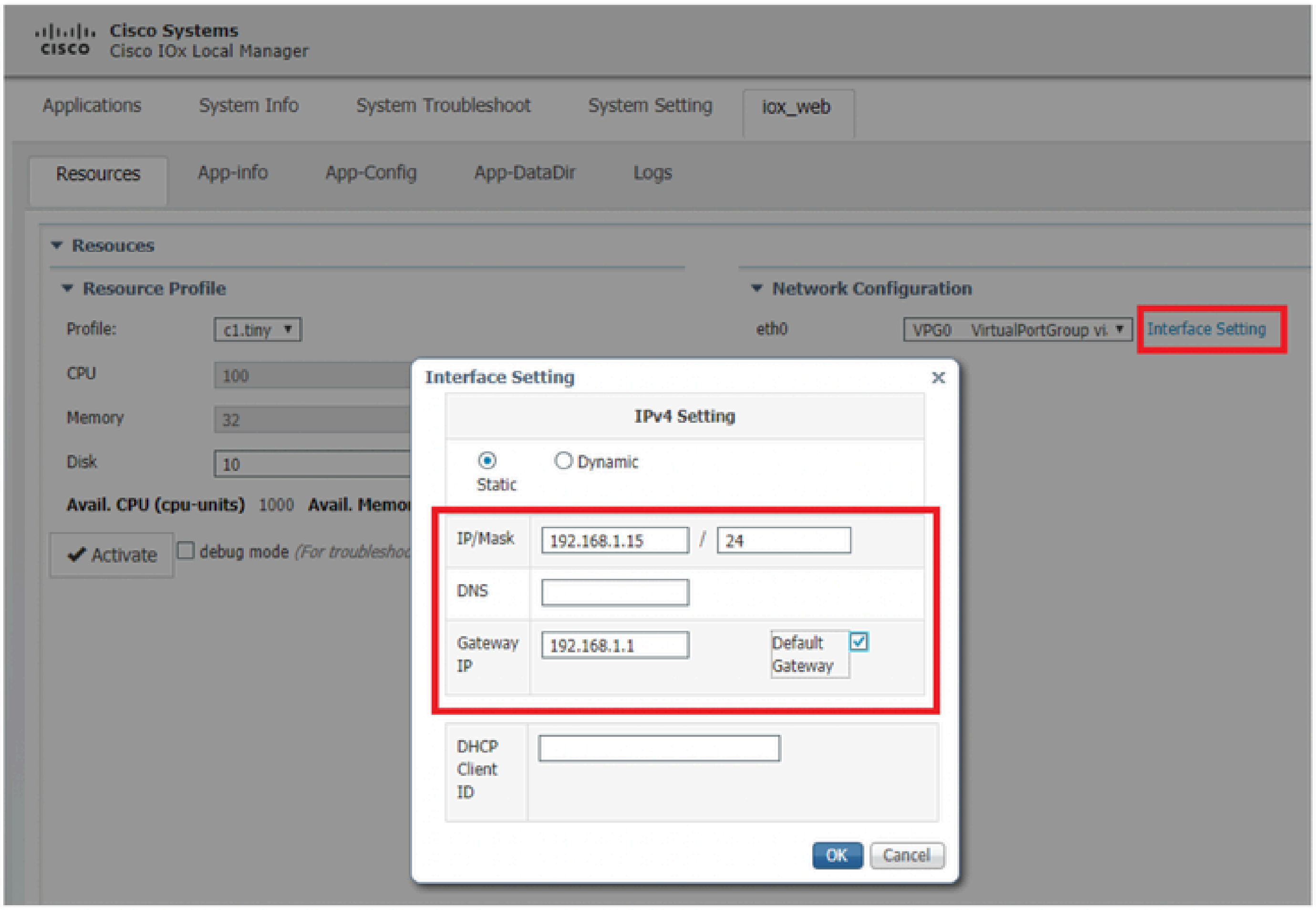


Application Id: iox_web
Select Application Archive package.tar

步驟 5.上傳封裝後，將其啟用，如下圖所示：



步驟 6.在Resources頁籤中，打開介面設定以指定要分配給應用程式的固定IP，如下圖所示：



Profile: c1.tiny
CPU 100
Memory 32
Disk 10
Avail. CPU (cpu-units) 1000 Avail. Memo
☐ debug mode *(For troublesho*
eth0 VPG0 VirtualPortGroup vi

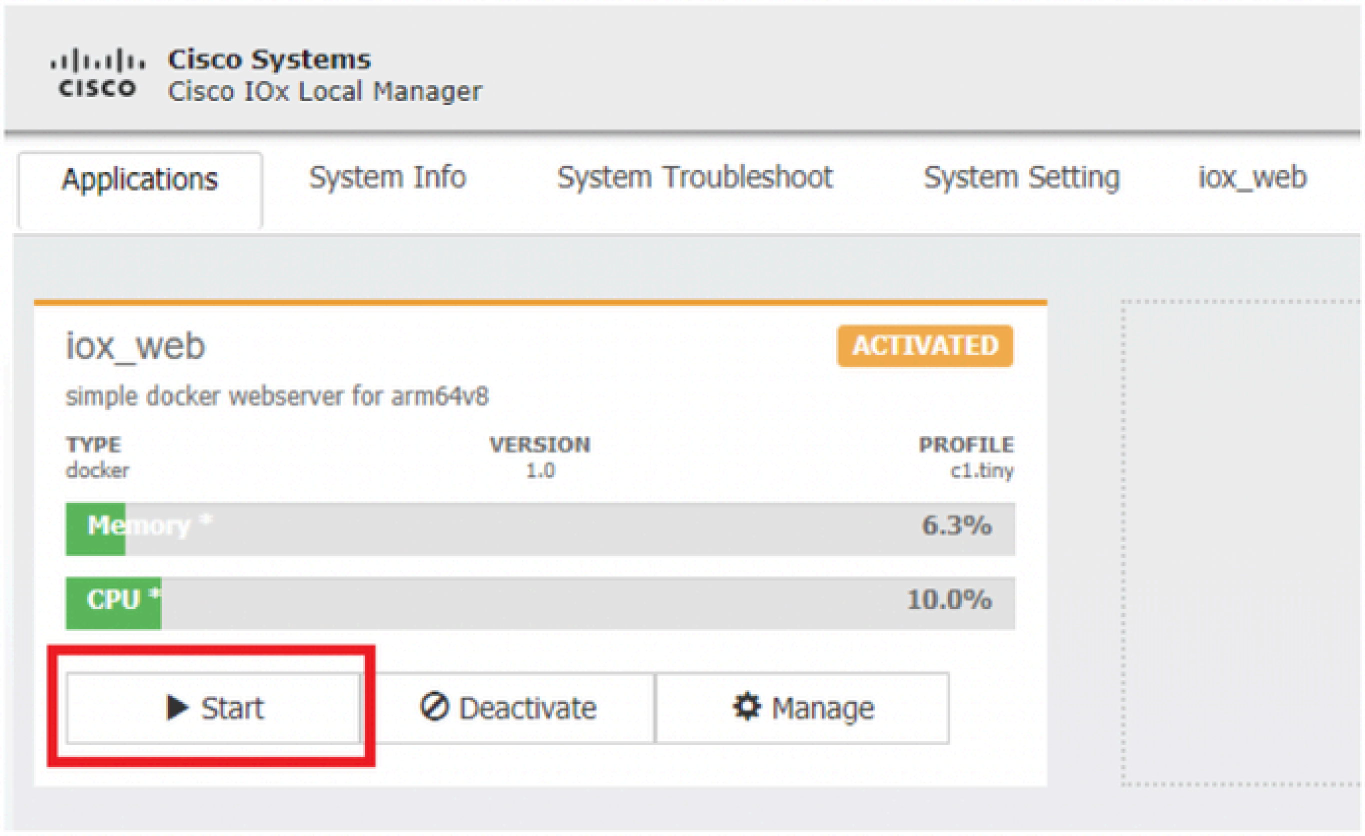
Interface Setting

IPv4 Setting

[✓ Static] Dynamic

IP/Mask	192.168.1.15 / 24	
DNS		
Gateway IP	192.168.1.1	Default Gateway ✓
DHCP Client ID		

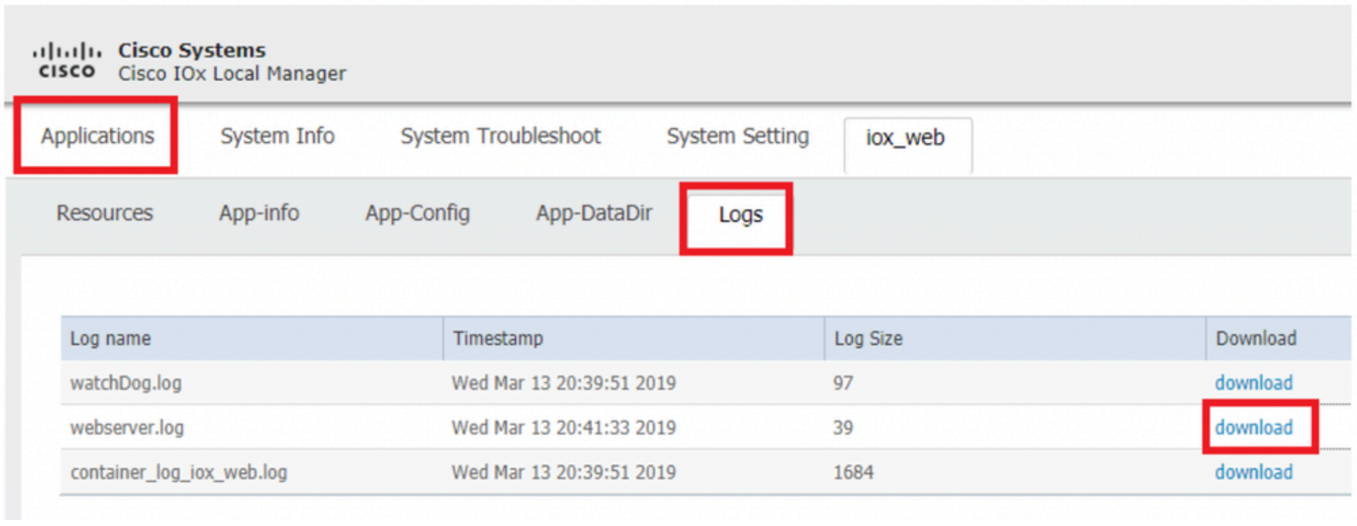
步驟 7.按一下OK，然後Activate。動作完成之後，導覽回主Local Manager頁面(頂端功能表上的Applications按鈕)，然後啟動應用程式，如下圖所示：



完成這些步驟後，您的應用程式即可開始執行。

疑難排解

為了對配置進行故障排除，請使用本地管理器檢查您在Python指令碼中建立的日誌檔案。導航到應用程式，按一下iox_web應用程式上的管理，然後選擇日誌頁籤，如圖所示：



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