

使用Vagrant和Virtualbox/VMWare构建IOx应用

目录

[简介](#)

[先决条件](#)

[Windows/ MAC Intel/ Linux](#)

[基于MAC ARM - M1/M2/M3](#)

[使用Vagrant建立环境的步骤](#)

[行动摘要](#)

[构建自定义IOx应用程序的过程](#)

[部署IOx应用](#)

[故障排除](#)

简介

本文档介绍如何使用Vagrant和Virtualbox构建IOx应用，以及如何在IOx本地管理器GUI中部署它们。

先决条件

Windows/ MAC Intel/ Linux

- Git
- 流浪者
- Virtualbox

基于MAC ARM - M1/M2/M3

- Git
- 流浪者
- VMWare融合
- vagrant-vmware-desktop插件

登录 Cisco.com。

- [流浪者](#)
- [VirtualBox](#)

使用Vagrant建立环境的步骤

行动摘要

- vagrantfile配置可设置基于其主机体系结构的虚拟机环境。
- 它根据架构将虚拟机配置为使用VMware Fusion或VirtualBox
- 它为虚拟机调配必要的软件和工具，包括QEMU(Quick EMUlator)、Docker和ioxclient。
- 配置会自动为amd64目标思科平台设备构建一个iperf应用示例。

步骤1:在本地系统中克隆Github存储库：

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

或者，将配置盘柜的内容复制并粘贴到“Vagrantfile”中。这将在本地系统中创建一个名为“Vagrantfile”的文件：

```
# -*- 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
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

```

第二步：确保取消对“export VER="5:24.0.9-1-ubuntu.22.04-jammy”行的注释，并注释所有其他导出语句。这对应于您想在此Vagrant环境中安装的Docker引擎版本：

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

第三步：使用Vagrantfile所在目录中的vagrant up命令启动Vagrant环境，并观察为amd64 tar文件成功构建的iperf IOx应用程序：

```
vagrant up
```

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

构建自定义IOx应用程序的过程

本节介绍如何使用流转环境构建自定义IOx应用。

注意：虚拟机中的目录“/vagrant”与主机系统中包含“Vagrantfile”的目录是同步的。

如图所示，new.js文件在VM内创建，也可以在主机系统上访问：

```
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 %
```

步骤1:将示例应用程序克隆到“Vagrantfile”所在的文件夹。此示例中使用“[iox-multiarch-nginx-nyancat-sample](#)”应用程序：

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

第二步：通过SSH登录迷走设备：

```
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:~$
```

第三步：构建应用程序：

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

构建过程完成后，您现在拥有两个可供部署的IOx应用(用于amd64的“[iox-amd64-nginx-nyancat-sample.tar.gz](#)”和用于目标平台的“[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 %
```

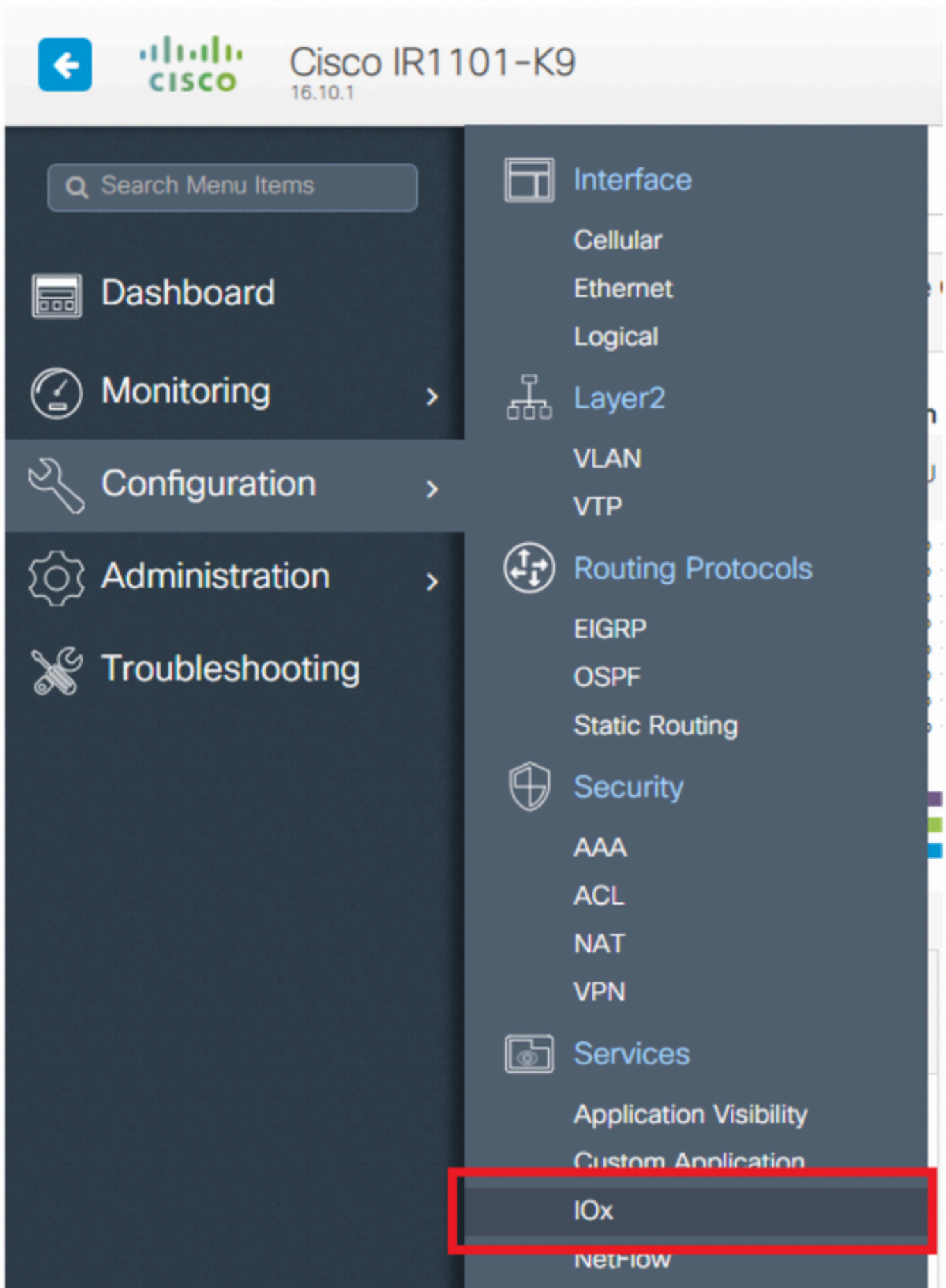
部署IOx应用

步骤1:使用Web界面访问IR1101 :

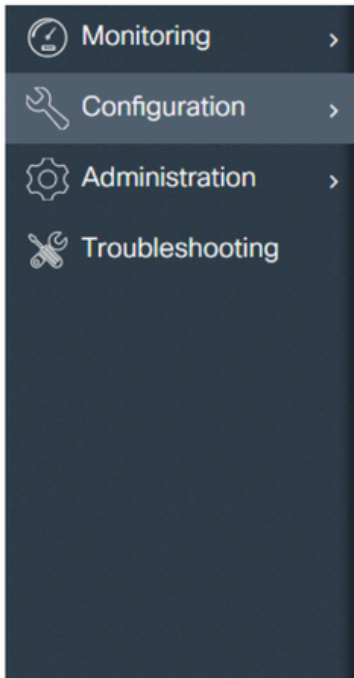


© 2005-2018 - Cisco Systems, Inc. All rights reserved. Cisco, the Cisco logo, and Cisco Systems are registered trademarks or trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries. All third party trademarks are the property of their respective owners.

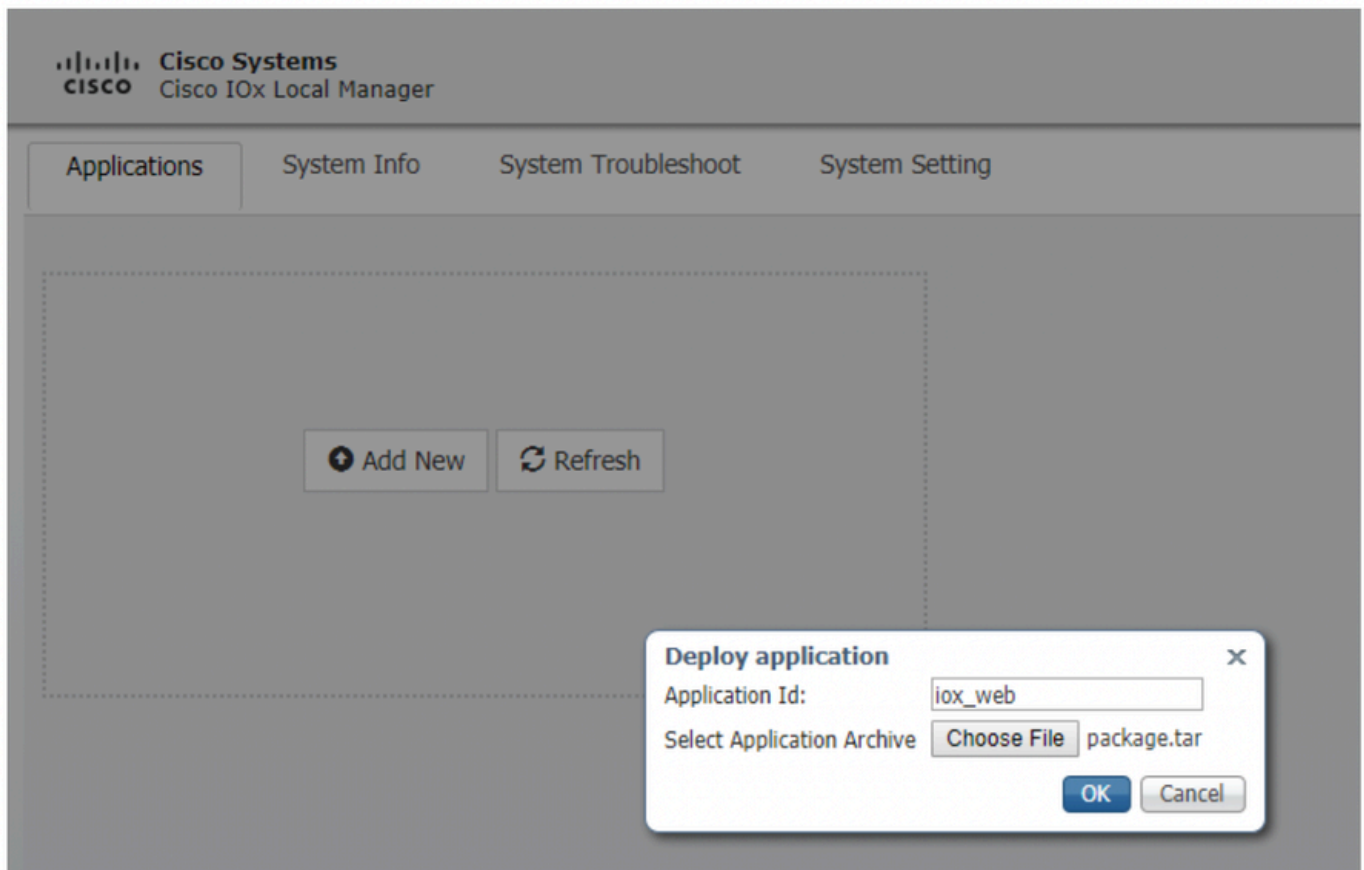
第二步：使用权限15帐户：



第三步：在IOx本地管理器登录中，使用相同的帐户继续，如图所示：



第四步：单击Add New，选择IOx应用程序的名称，并选择Procedure to Set Up Environment Using Vagrant部分步骤3中构建的package.tar，如图所示：



第五步：上传软件包后，将其激活，如图所示：

Applications

System Info

System Troubleshoot

System Setting

iox_web

DEPLOYED

simple docker webserver for arm64v8

TYPE	VERSION	PROFILE
docker	1.0	c1.tiny

Memory *

6.3%

CPU *

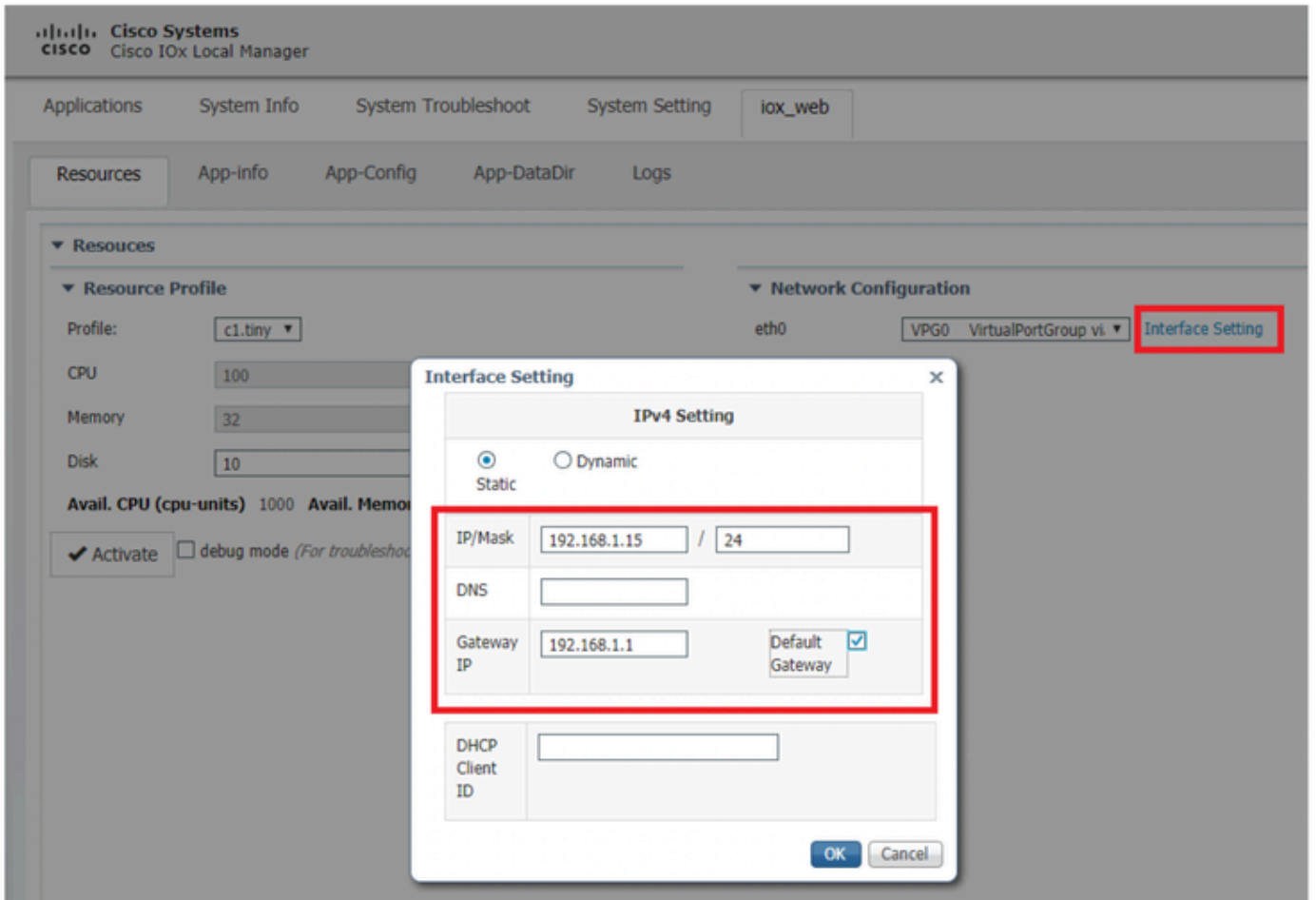
10.0%

✓ Activate

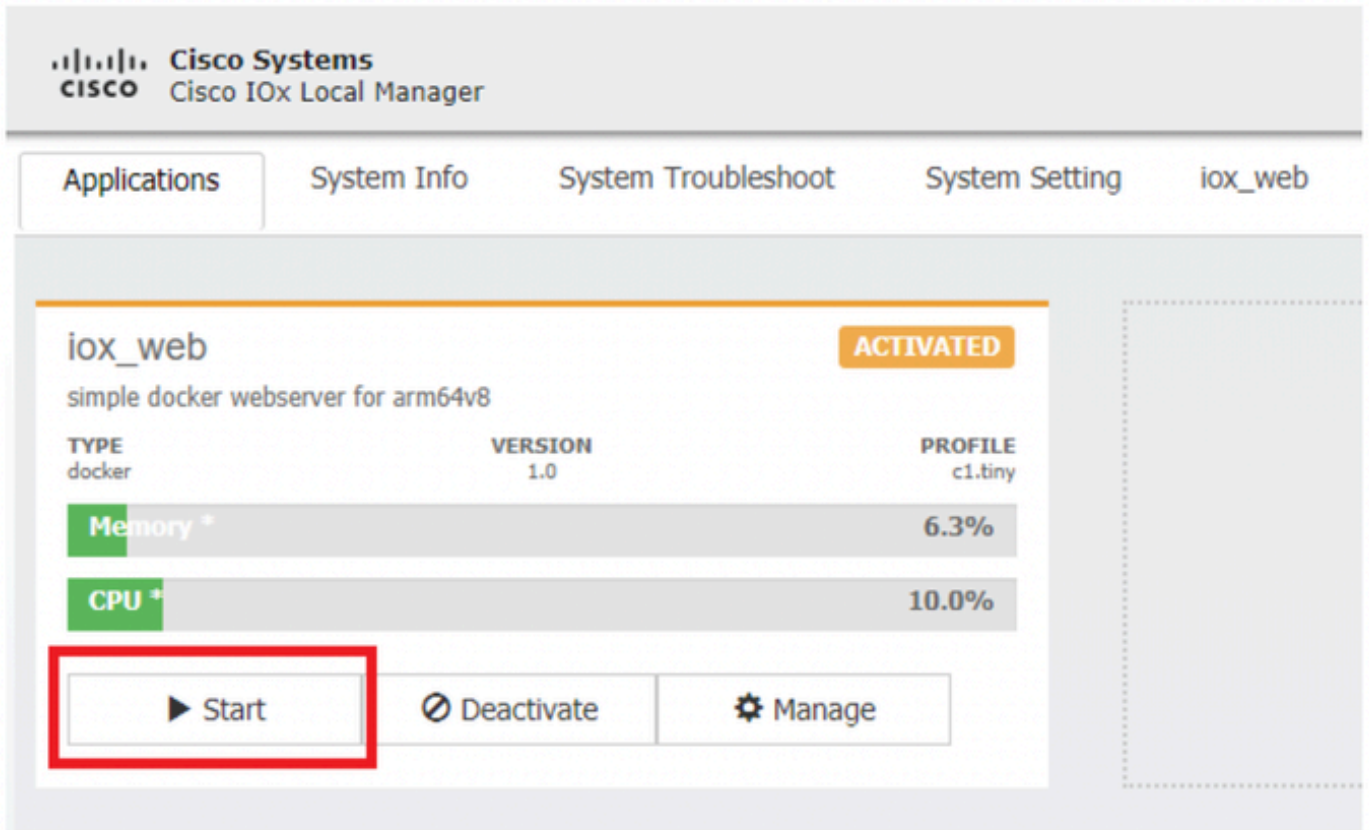
Upgrade

Delete

第六步：在Resources选项卡中，打开接口设置以指定要分配给应用的固定IP，如图所示：



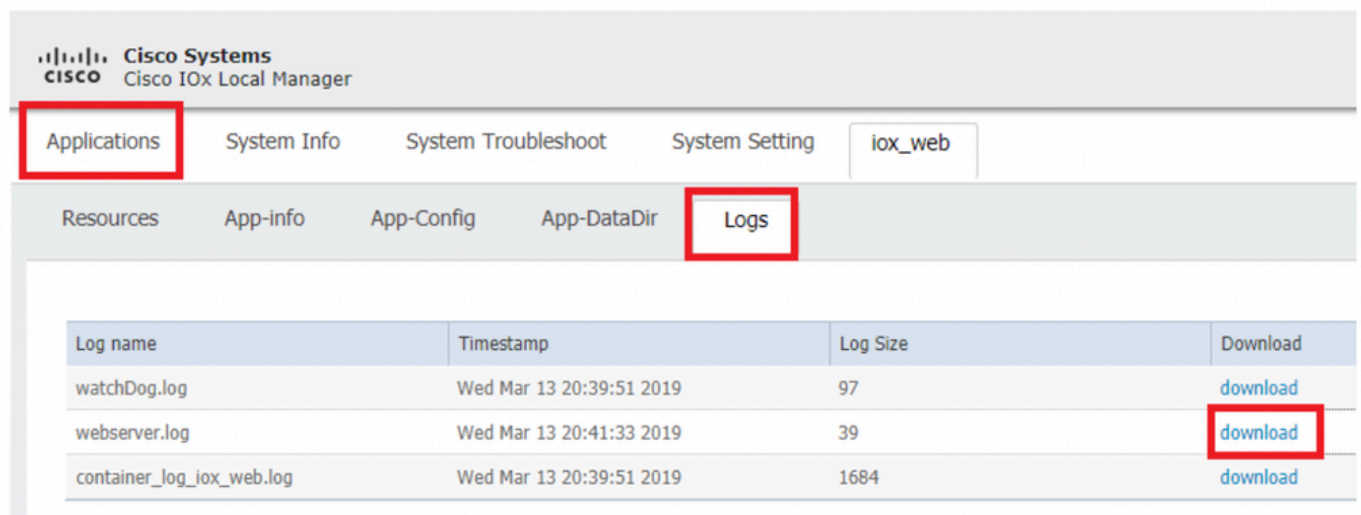
步骤 7.单击OK，然后单击Activate。操作完成后，导航回主Local Manager页面(顶部菜单上的Applications按钮)，然后启动应用程序，如图所示：



完成这些步骤后，您的应用程序即可开始运行。

故障排除

为了对配置进行故障排除，请使用本地管理器检查您在Python脚本中创建的日志文件。导航到应用，点击iox_web应用上的管理，然后选择日志选项卡，如图所示：



关于此翻译

思科采用人工翻译与机器翻译相结合的方式将此文档翻译成不同语言，希望全球的用户都能通过各自的语言得到支持性的内容。

请注意：即使是最好的机器翻译，其准确度也不及专业翻译人员的水平。

Cisco Systems, Inc. 对于翻译的准确性不承担任何责任，并建议您总是参考英文原始文档（已提供链接）。