Cree aplicaciones IOx con Vagrant y Virtualbox/VMWare

Contenido

Introducción

Prerequisites

- Windows/ MAC Intel/ Linux
- Basado en MAC ARM: M1/M2/M3

Procedimiento para configurar un entorno de compilación mediante Vagrant

- Resumen de acciones

Procedimiento para crear una aplicación IOx personalizada

Implementación de la aplicación IOx

Troubleshoot

Introducción

Este documento describe cómo crear aplicaciones IOx usando Vagrant y Virtualbox e implementarlas en la GUI del administrador local de IOx.

Prerequisites

Windows/ MAC Intel/ Linux

- Git
- Vagabundo
- Virtualbox

Basado en MAC ARM: M1/M2/M3

- Git
- Vagabundo
- VMWare Fusion
- complemento vagrant-vmware-desktop

Para descargar:

- Vagabundo
- VirtualBox

Procedimiento para configurar un entorno de compilación

mediante Vagrant

Resumen de acciones

- La configuración de vagrantfile configura un entorno de VM basado en su arquitectura de máquina host.
- Configura la máquina virtual para utilizar VMware Fusion o VirtualBox, en función de la arquitectura
- Aprovisiona la máquina virtual con el software y las herramientas necesarias, incluidos QEMU (Quick EMUlator) , Docker y ioxclient.
- La configuración crea automáticamente una aplicación iperf de ejemplo para los dispositivos de la plataforma Cisco de destino amd64.

Paso 1. Clonar el repositorio de Github en su sistema local:

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

Como alternativa, copie y pegue el contenido del contenedor de configuración en "Vagrantfile". Esto crea un archivo con el nombre "Vagrantfile" en el sistema local:

```
# -*- 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 .
```

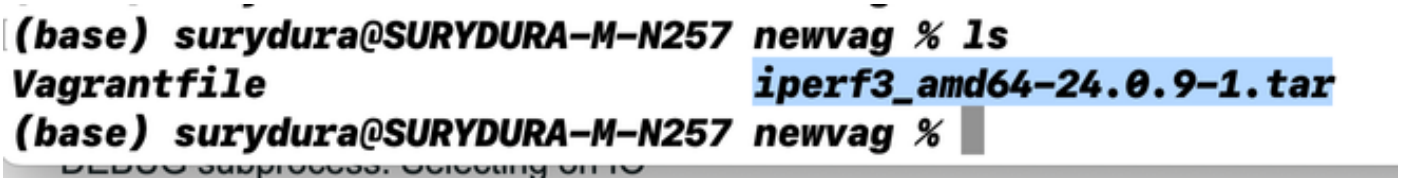
```
    cp package.tar /vagrant/iperf3_amd64-$(echo $VER | pcregrep -o1 ':([0-9.-]+)~').tar
  SHELL
end
```

Paso 2. Asegúrese de que la línea "export VER="5:24.0.9-1~ubuntu.22.04~jammy" no tenga comentarios y que todas las demás sentencias de exportación estén comentadas. Esto corresponde a la versión de Docker Engine que desea instalar en este entorno Vagrant:

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

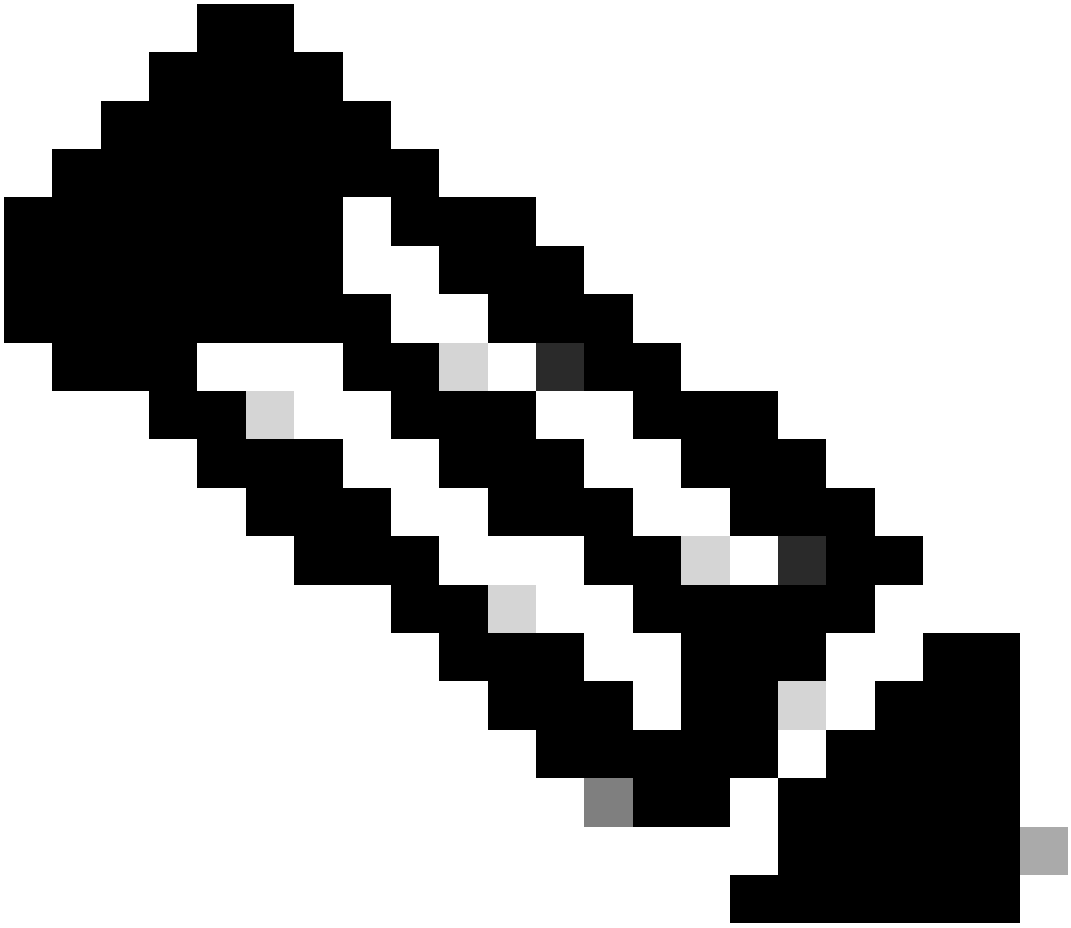
Paso 3. Inicie el entorno Vagrant con el comando vagrant up en el directorio donde reside el archivo Vagrant y observe una compilación exitosa de la aplicación IOSx iperf para el archivo tar amd64:

```
vagrant up
```



Procedimiento para crear una aplicación IOx personalizada

En esta sección se describe cómo crear una aplicación IOx personalizada utilizando el entorno Vagrant.



Nota: El directorio "/vagrant" en la VM y el directorio que contiene el "Vagrant file" en el sistema host están sincronizados.

Como se muestra en la imagen, el archivo new.js se crea dentro de la VM y también es accesible en el sistema host:

```
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 %
```

Paso 1. Clonar una aplicación de ejemplo en la misma carpeta en la que se encuentra "Vagrantfile". En este ejemplo, se utiliza la aplicación "iox-multiarch-nginx-nyancat-sample":

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

Paso 2. SSH en la máquina vagabunda:

```
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:~$
```

Paso 3. Genere la aplicación:

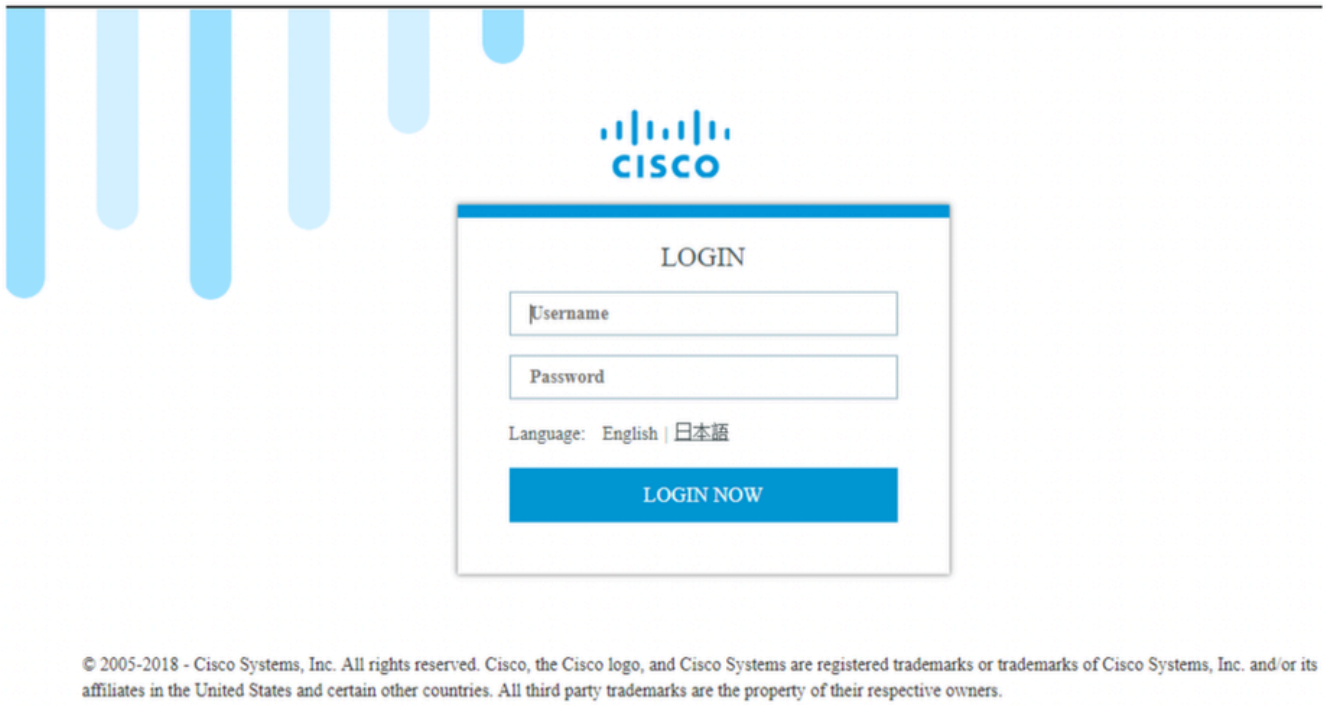
```
cd /vagrant/iox-multiarch-nginx-nyancat-sample/
chmod +x build
sh ./build
```

Una vez finalizado el proceso de creación, tendrá dos aplicaciones IOx listas para su implementación ("iox-amd64-nginx-nyancat-sample.tar.gz" para amd64 y "iox-arm64-nginx-nyancat-sample.tar.gz" para las plataformas de destino):

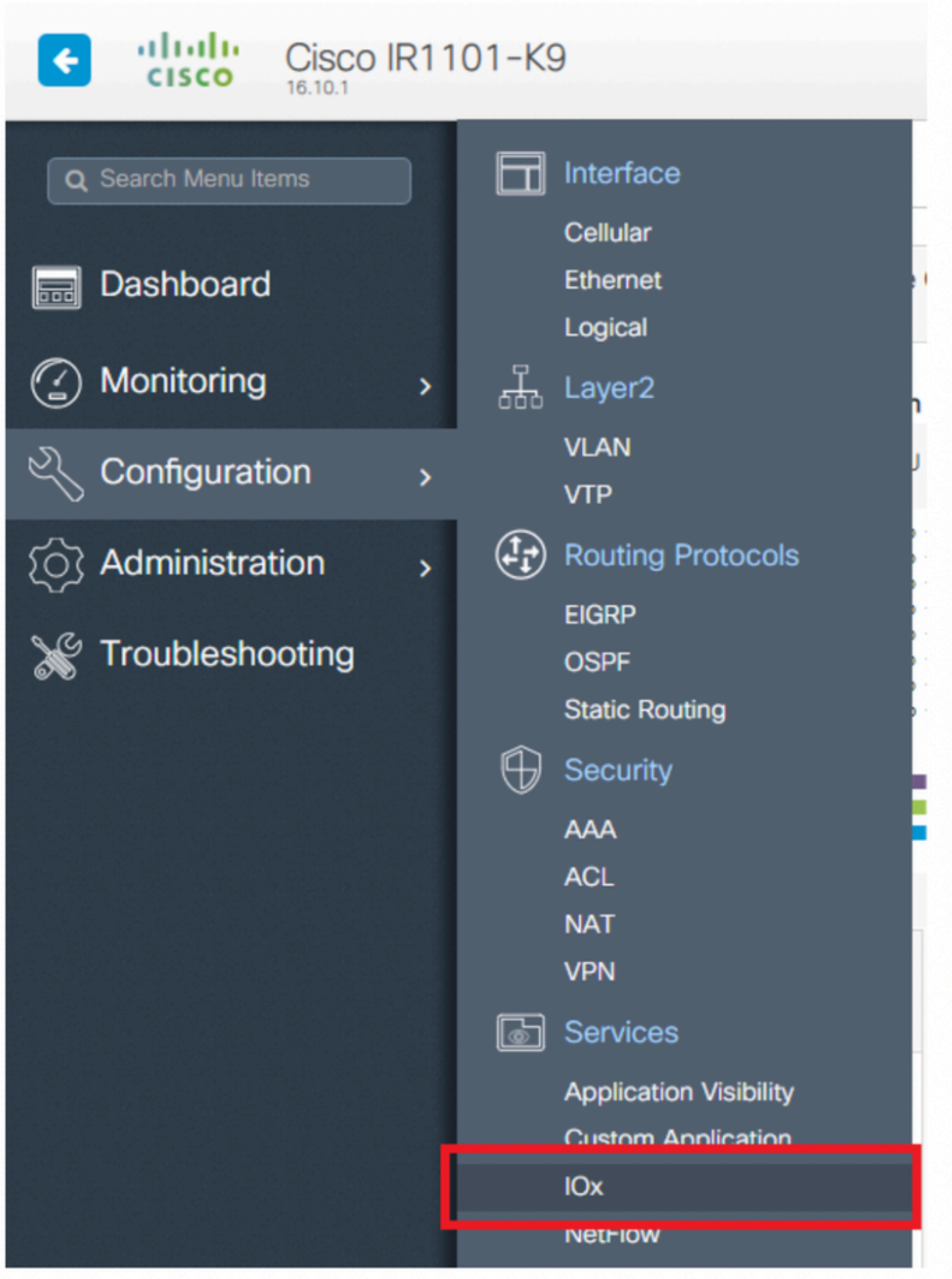
```
Package docker image iox-arm64-nginx-nyancat-sample at /vagrant/iox-multiarch-nginx-nyancat-sample/iox-arm64-nginx-nyancat-sample.tar.gz
vagrant@vagrant:/vagrant/iox-multiarch-nginx-nyancat-sample$ ls
Dockerfile  README.md  images                        iox-arm64-nginx-nyancat-sample.tar.gz  nyan-cat      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 %
```

Implementación de la aplicación IOx

Paso 1. Acceda al IR1101 mediante la interfaz web:



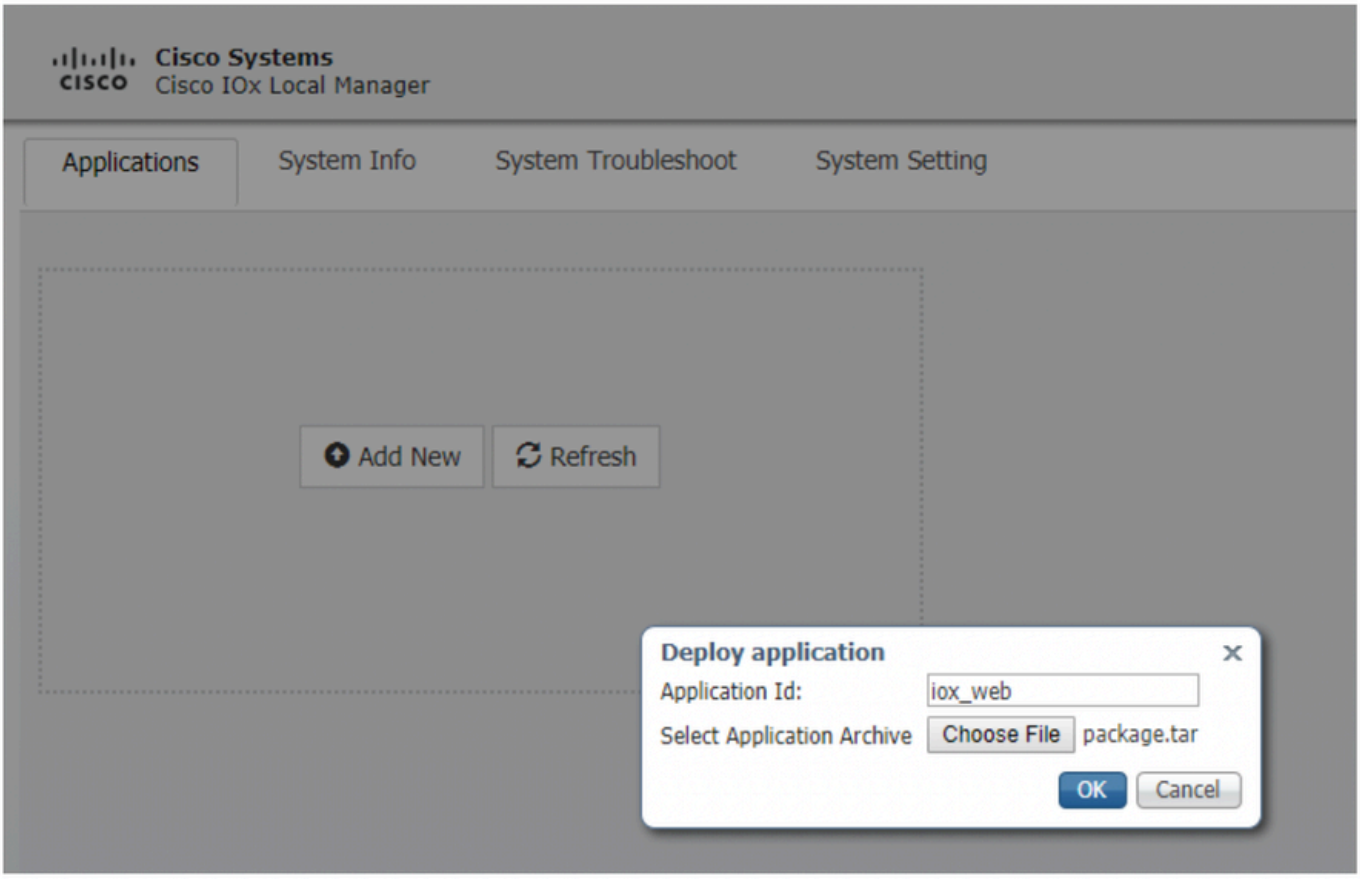
Paso 2. Utilice la cuenta de privilegio 15:



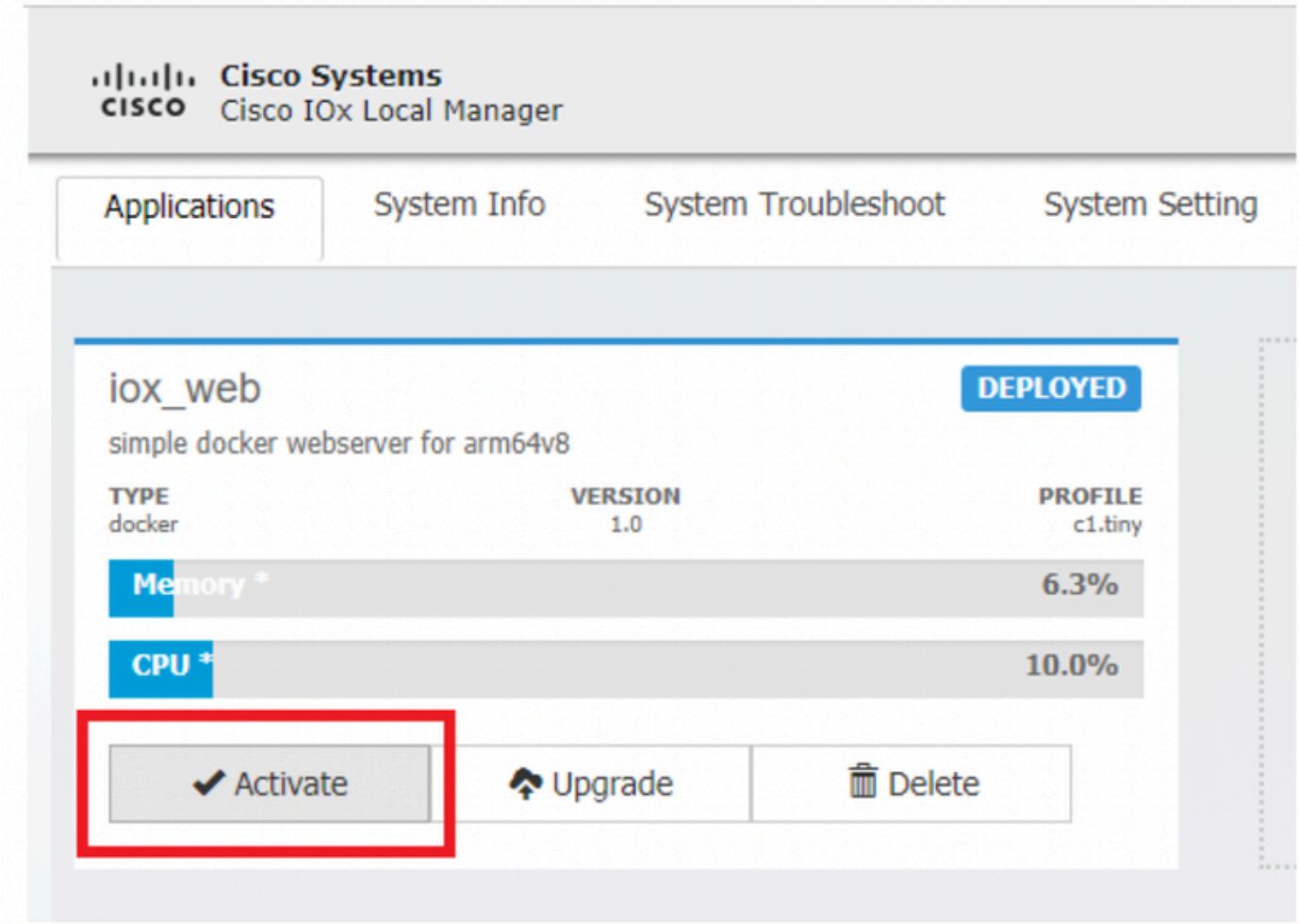
Paso 3. En el inicio de sesión de IOx Local Manager, utilice la misma cuenta para continuar como

se muestra en la imagen:

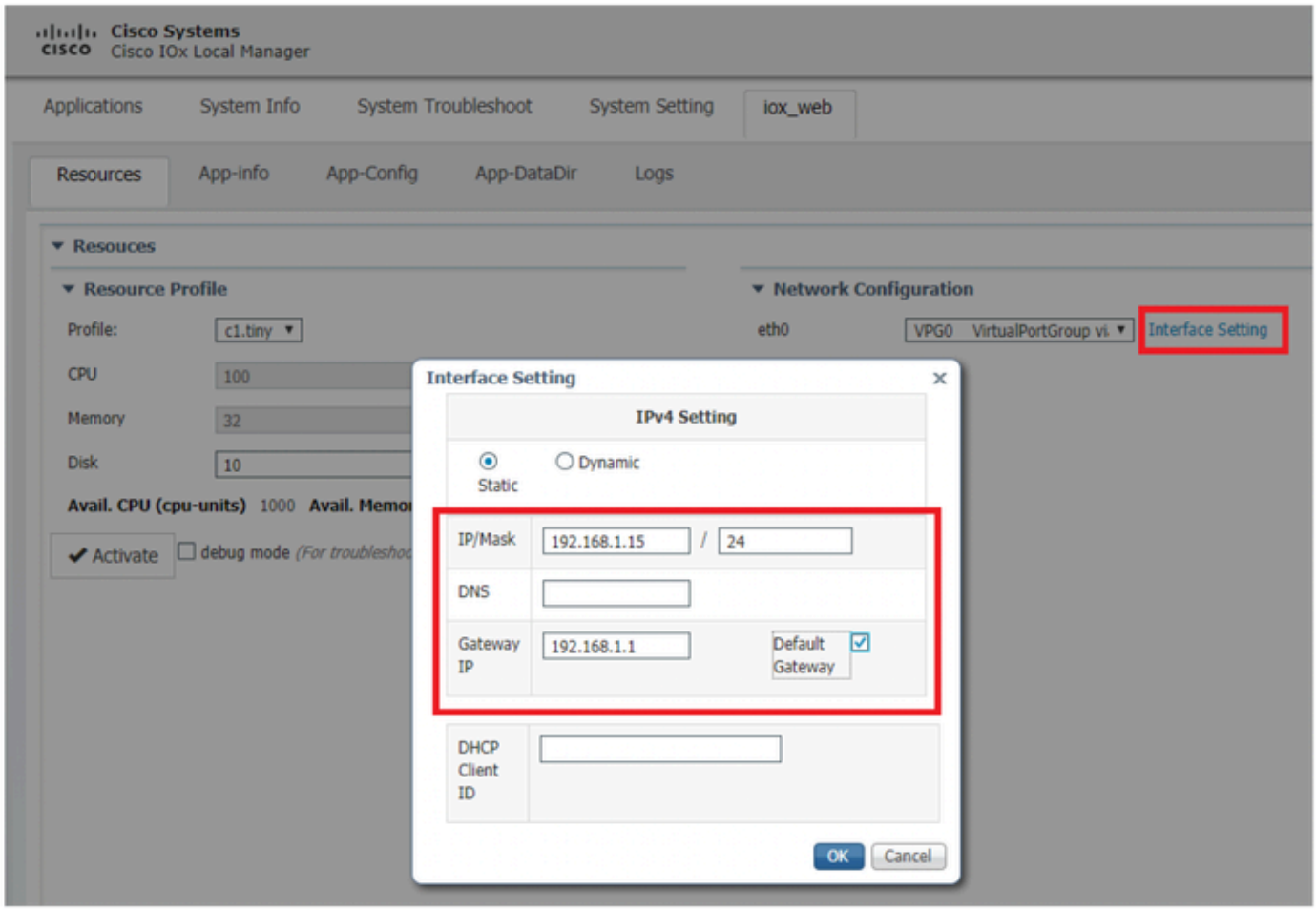
Paso 4. Haga clic en Add New, seleccione un nombre para la aplicación IOx y elija el package.tar que se generó en el Paso 3 de la sección Procedimiento para Configurar un Entorno de Compilación Usando Vagrant, como se muestra en la imagen:



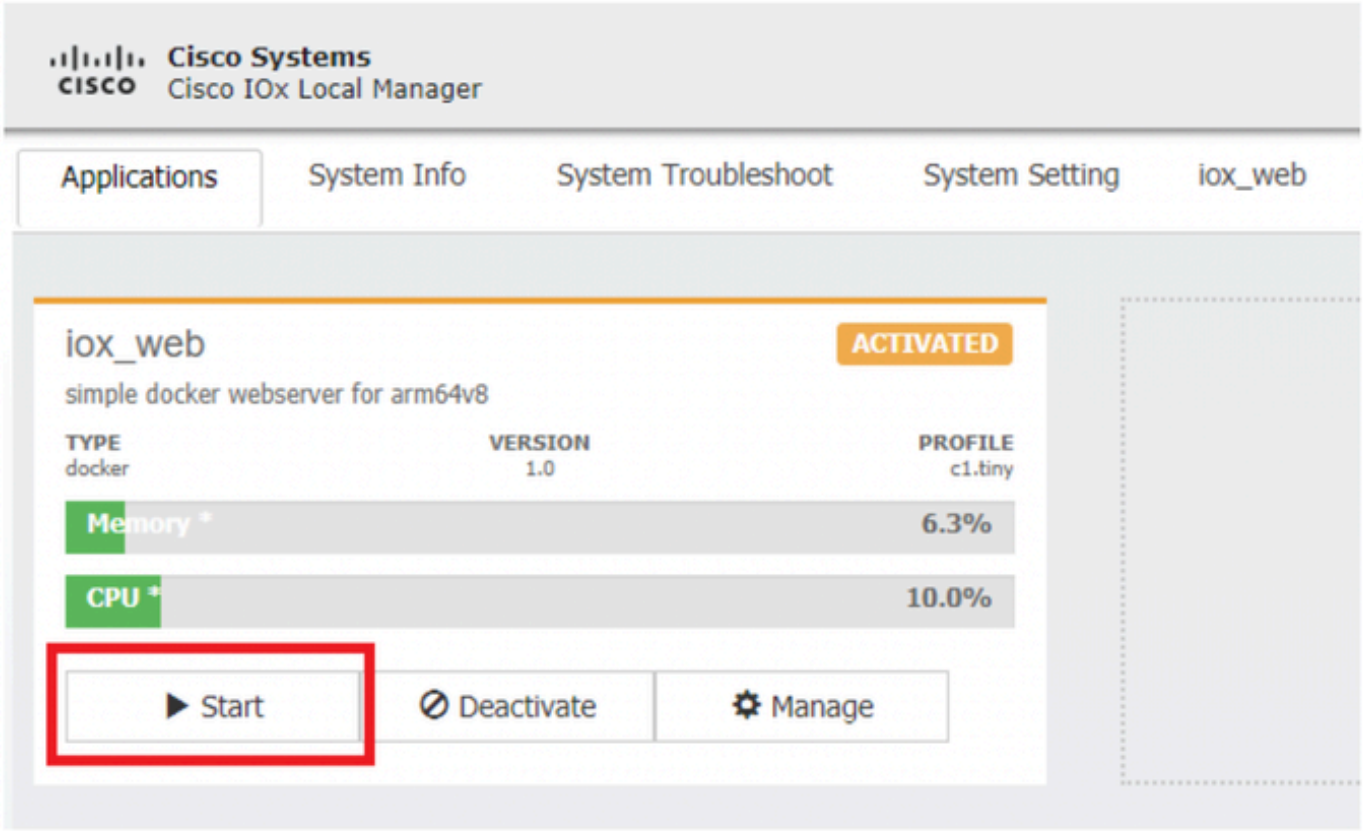
Paso 5. Una vez cargado el paquete, actívelo como se muestra en la imagen:



Paso 6. En la pestaña Resources, abra la configuración de la interfaz para especificar la IP fija que desea asignar a la aplicación como se muestra en la imagen:



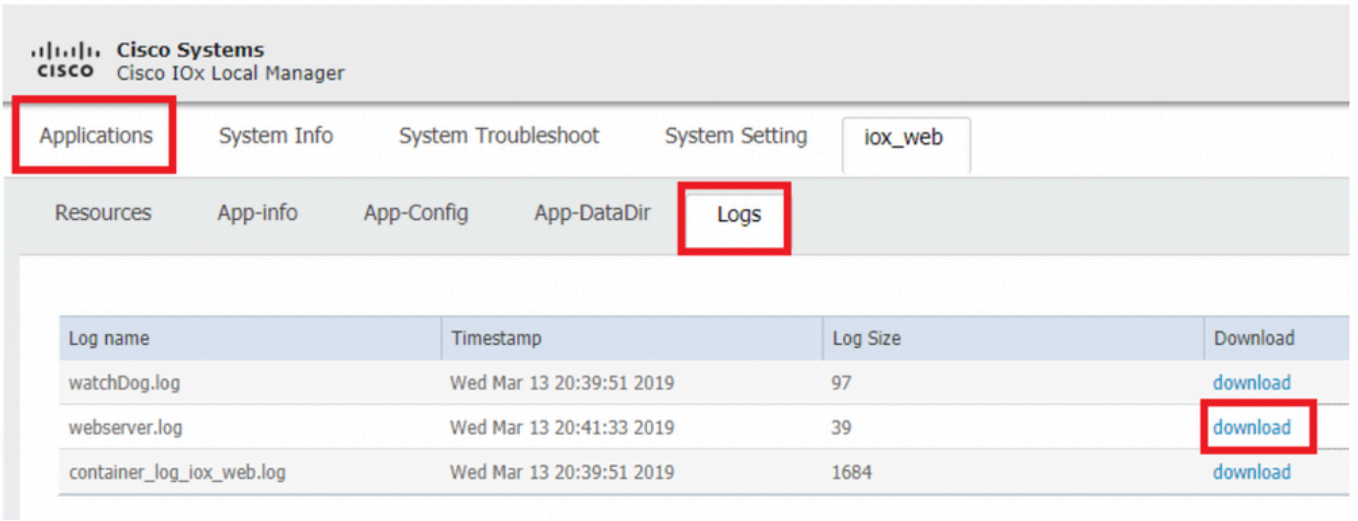
Paso 7. Haga clic en Aceptar, luego en Activar. Una vez finalizada la acción, vuelva a la página principal de Local Manager (botón Applications en el menú superior) y, a continuación, inicie la aplicación como se muestra en la imagen:



Después de realizar estos pasos, la aplicación está lista para ejecutarse.

Troubleshoot

Para resolver problemas de configuración, verifique el archivo de registro que creó en el script Python usando un administrador local. Navegue hasta Aplicaciones, haga clic en Administrar en la aplicación iox_web, luego seleccione la pestaña Registros como se muestra en la imagen:



Acerca de esta traducción
Cisco ha traducido este documento combinando la traducción automática y los recursos humanos a fin de ofrecer a nuestros usuarios en todo el mundo contenido en su propio idioma.

Tenga en cuenta que incluso la mejor traducción automática podría no ser tan precisa como la proporcionada por un traductor profesional.

Cisco Systems, Inc. no asume ninguna responsabilidad por la precisión de estas traducciones y recomienda remitirse siempre al documento original escrito en inglés (insertar vínculo URL).