Crie aplicativos IOx com Vagrant e Virtualbox/VMWare

Contents

Introdução

Este documento descreve como criar aplicativos IOx usando Vagrant e Virtualbox e implantá-los na GUI do gerenciador local do IOx.

Pré-requisitos

Windows/ MAC Intel/ Linux

- Git
- Vagante
- Caixa virtual

Baseado em MAC ARM - M1/M2/M3

- Git
- Vagante
- Fusão VMWare
- plugin vagrant-vmware-desktop

Para fazer o download:

- Vagante
- VirtualBox

Procedimento para Configurar o Ambiente de Compilação

Usando Vagrant

Resumo das ações

- A configuração do vagrantfile configura um ambiente de VM com base na arquitetura da máquina host.
- Ele configura a VM para usar o VMware Fusion ou o VirtualBox, dependendo da arquitetura
- Ele provisiona a VM com software e ferramentas necessários, incluindo QEMU (Quick EMUlator) , Docker e ioxclient.
- A configuração cria automaticamente um aplicativo iperf de exemplo para dispositivos de plataforma Cisco de destino amd64.

Etapa 1. Clonar o repositório Github em seu sistema local:

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

Como alternativa, copie e cole o conteúdo do compartimento de configuração em "Vagrantfile". Isso cria um arquivo com o nome "Vagrantfile" no 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
```

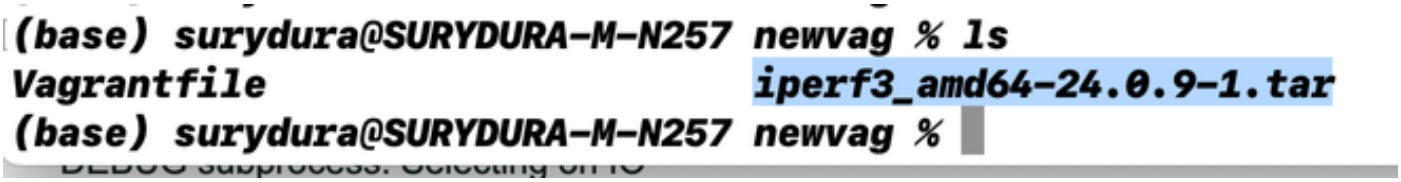
```
  SHELL
end
```

Etapa 2. Certifique-se de que a linha "export VER="5:24.0.9-1~ubuntu.22.04~jammy" não esteja comentada e todas as outras instruções de exportação sejam comentadas. Isso corresponde à versão do Docker Engine que você deseja instalar neste ambiente Vagrant:

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

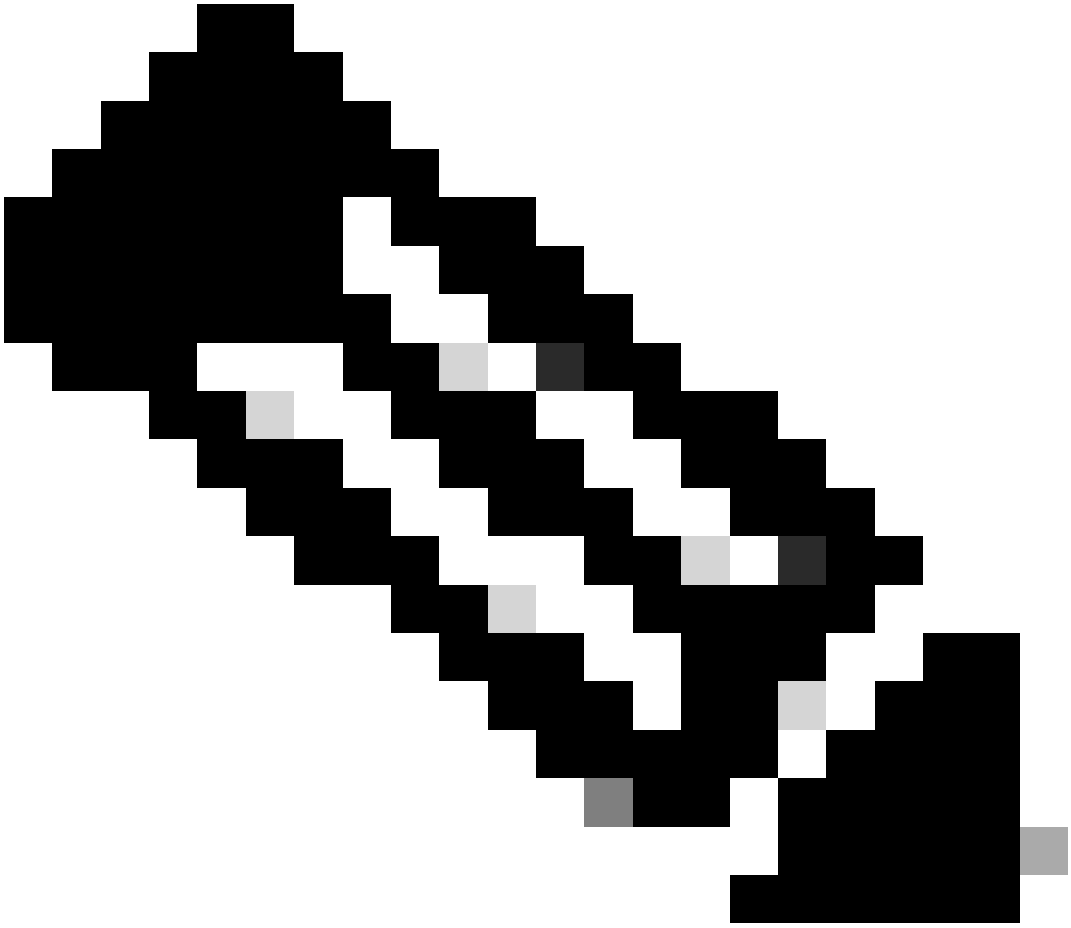
Etapa 3. Inicie o ambiente Vagrant com o comando vagrant up no diretório em que reside o arquivo Vagrantfile e observe uma compilação bem-sucedida do aplicativo iperf IOx para o arquivo tar amd64:

```
vagrant up
```



Procedimento para criar um aplicativo IOx personalizado

Esta seção descreve como criar um aplicativo IOx personalizado usando o ambiente vagrant.



Observação: o diretório "/vagrant" na VM e o diretório que contém o "Vagrantfile" no sistema host estão em sincronia.

Como mostrado na imagem, o arquivo new.js é criado dentro da VM e também pode ser acessado no 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 %
```

Etapa 1. Clonar um aplicativo de amostra para a mesma pasta onde o "Vagrantfile" reside. Neste exemplo, o aplicativo "iox-multiarch-nginx-nyancat-sample" é usado:

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

Etapa 2. SSH na 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:~$
```

Etapa 3. Construir o aplicativo:

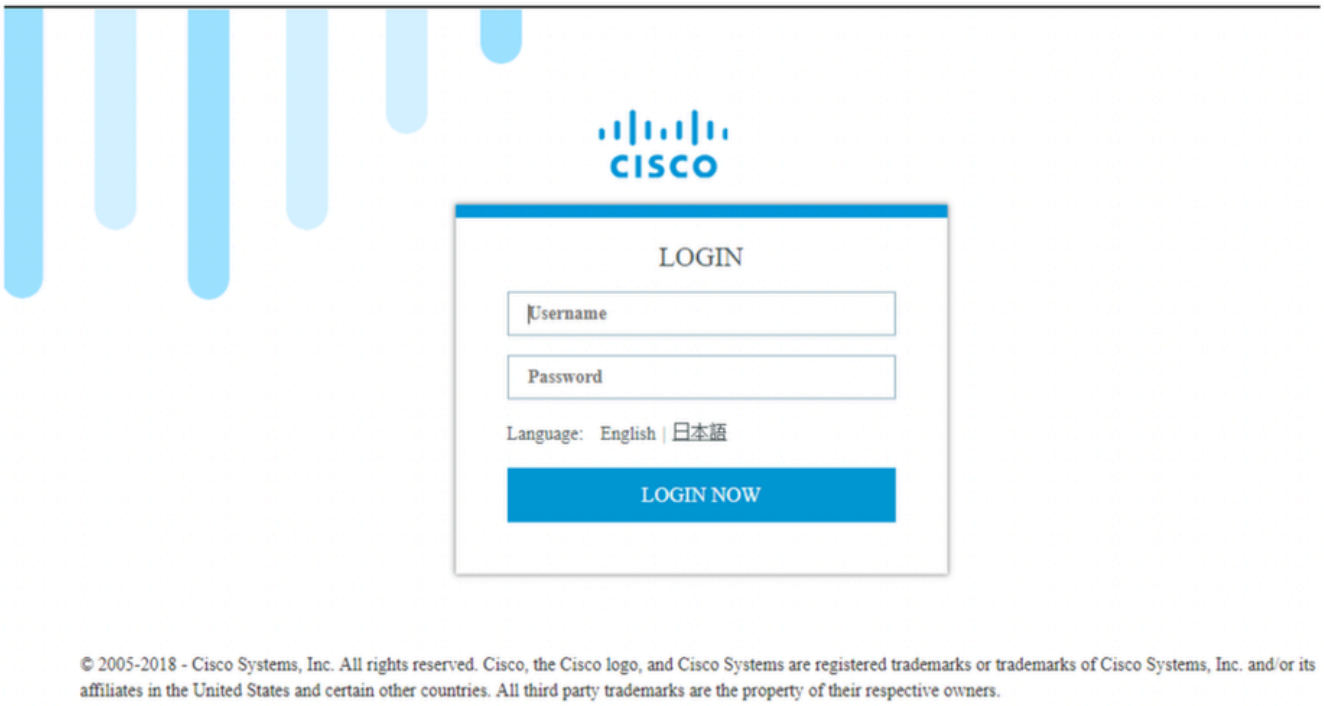
```
cd /vagrant/iox-multiarch-nginx-nyancat-sample/
chmod +x build
sh ./build
```

Após a conclusão do processo de criação, agora você tem dois aplicativos IOx prontos para implantação ("iox-amd64-nginx-nyancat-sample.tar.gz" para amd64 e "iox-arm64-nginx-nyancat-sample.tar.gz" para plataformas de destino):

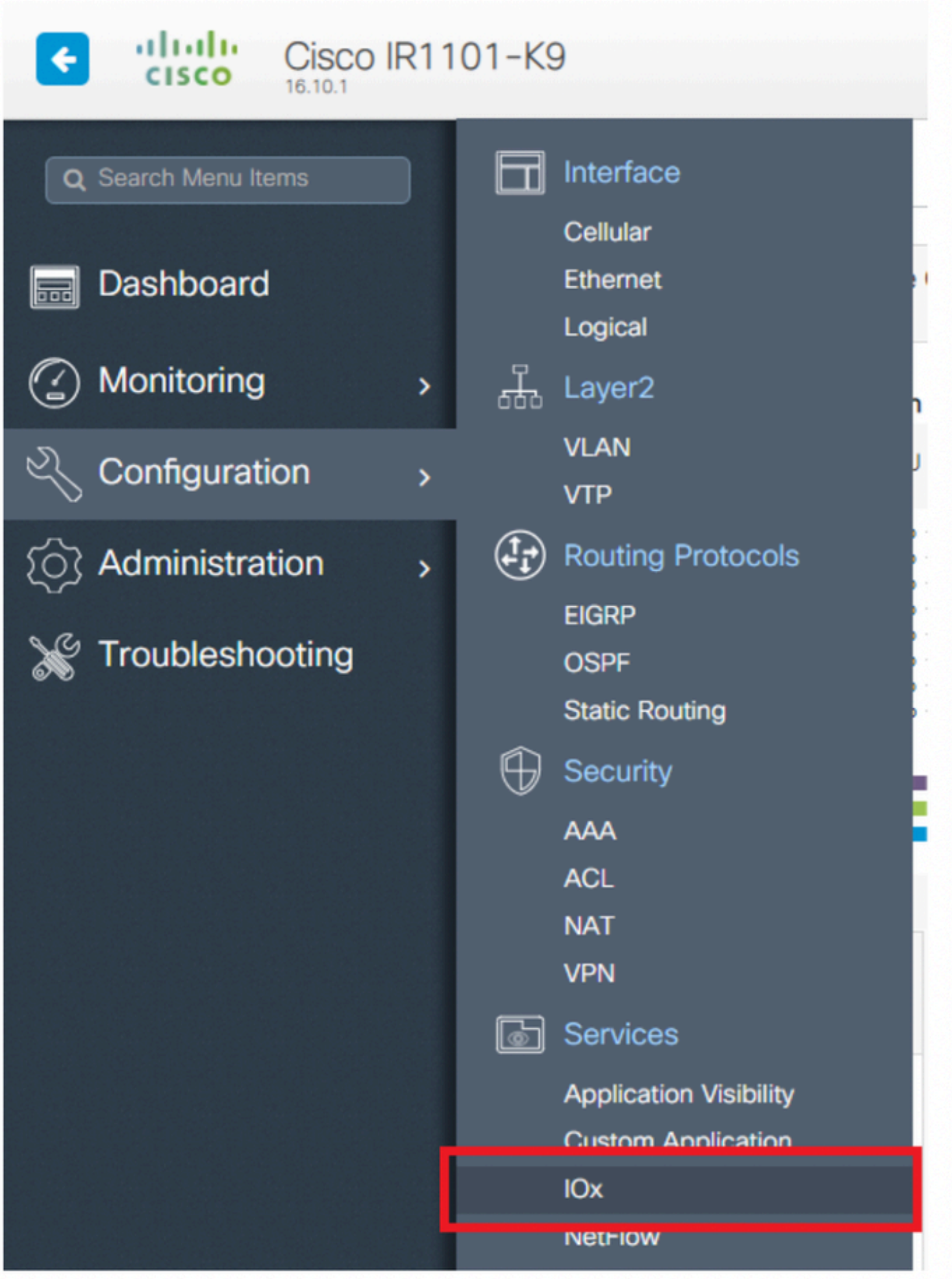
```
Package docker image iox-arm64-nginx-nyancat-sample at /vagrant/iox-multiarch-nginx-nyancat-sample/iox-arm64-nginx-nyancat-sample.tar.gz
vagrant@vagrant:/vagrant/iox-multiarch-nginx-nyancat-sample$ ls
Dockerfile  README.md  images                        iox-arm64-nginx-nyancat-sample.tar.gz  nyan-cat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

Implantar o aplicativo IOx

Etapa 1. Acesse o IR1101 com o uso da interface da Web:



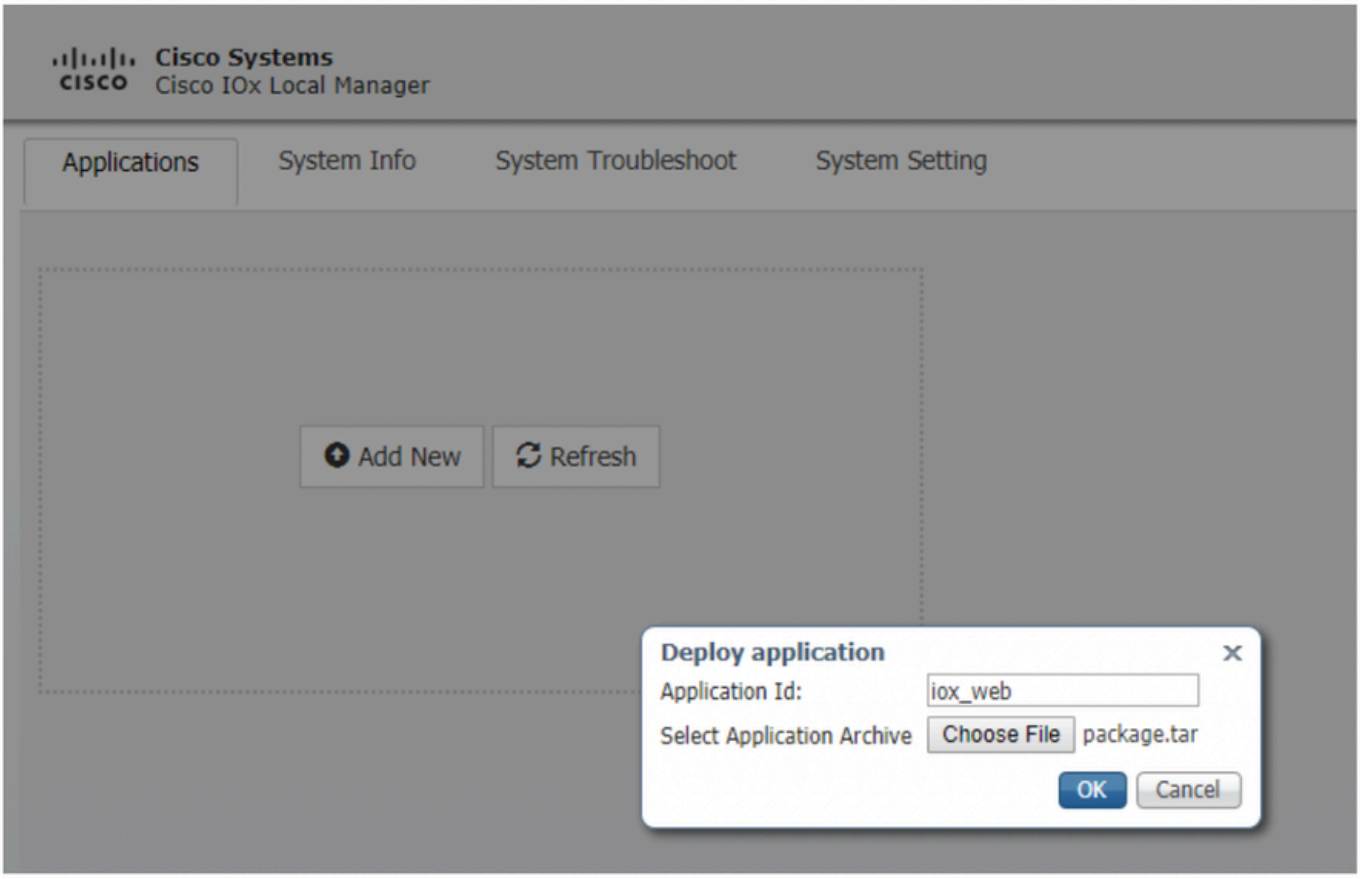
Etapa 2. Use a conta do privilégio 15:



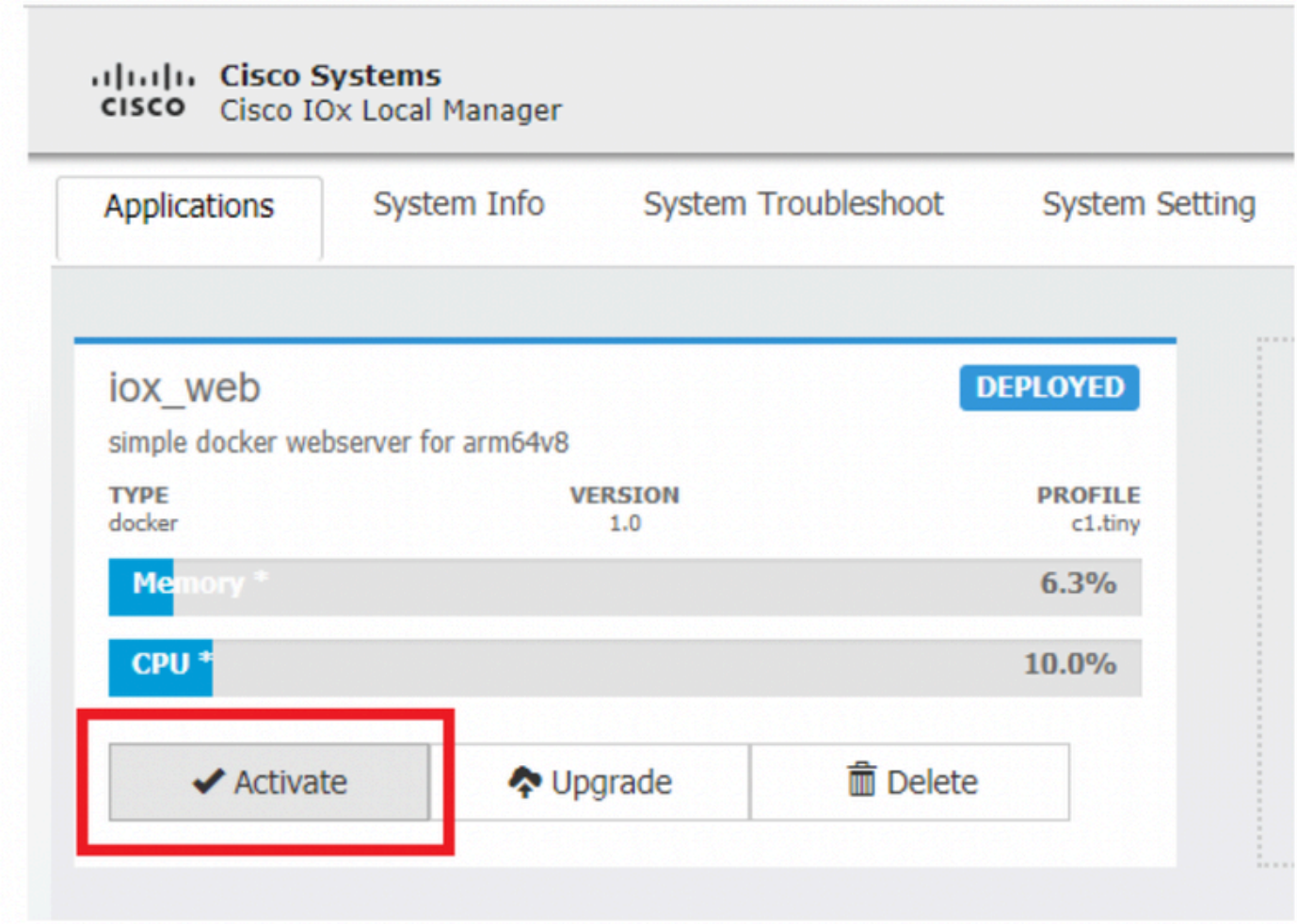
Etapa 3. No login do IOx Local Manager, use a mesma conta para continuar como mostrado na

imagem:

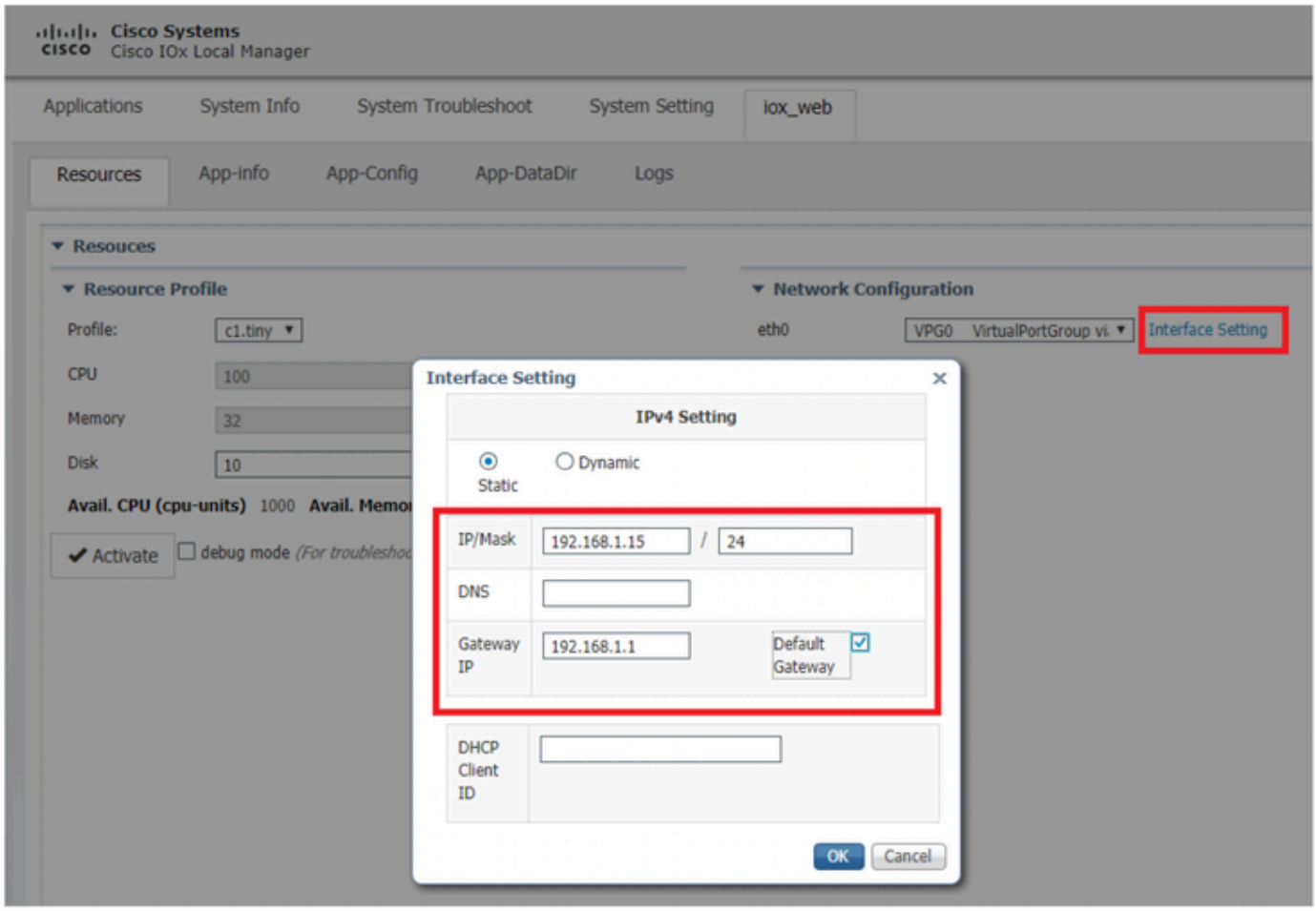
Etapa 4. Clique em Add New, selecione um nome para o aplicativo IOx e escolha o package.tar que foi criado na Etapa 3 da seção Procedure to Set Up Build Environment Using Vagrant , como mostrado na imagem:



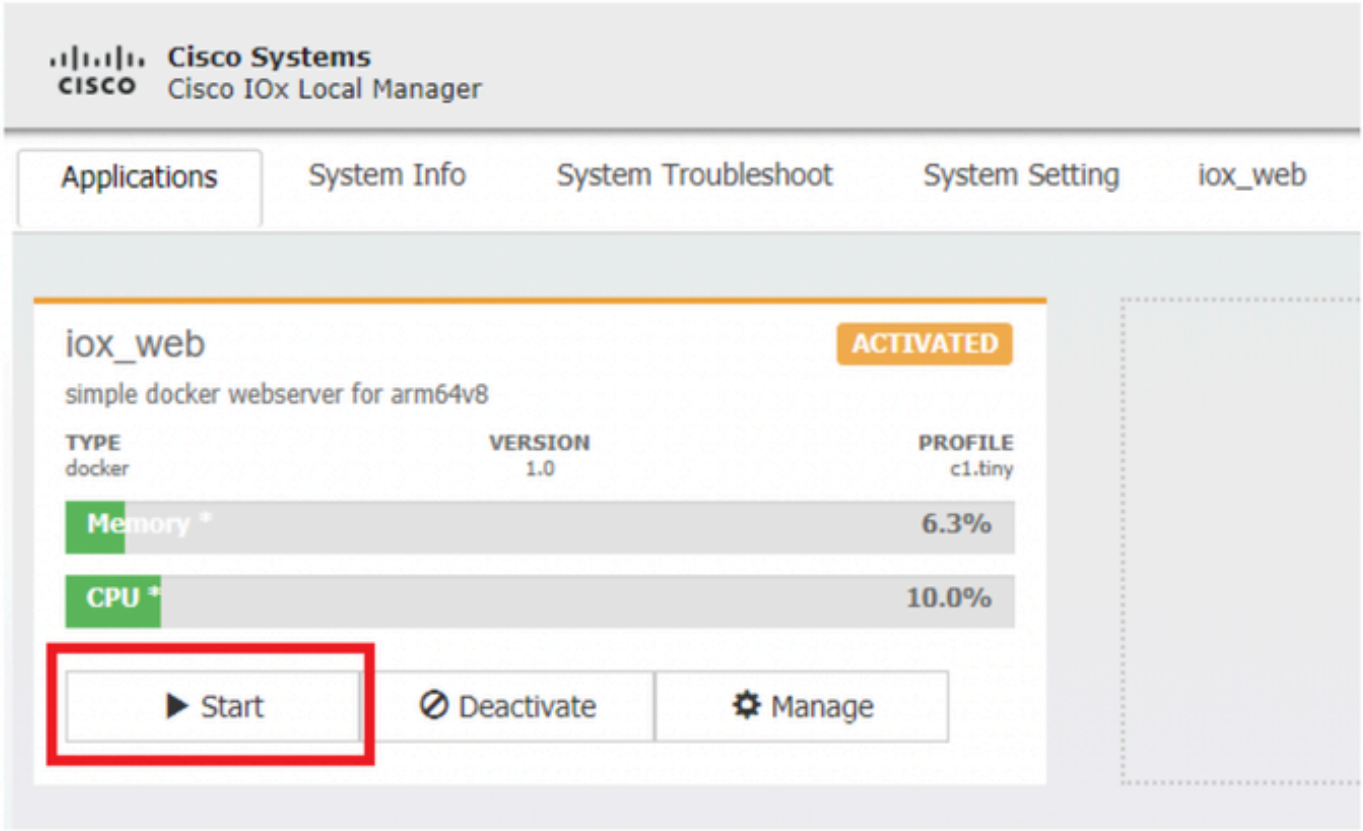
Etapa 5. Depois que o pacote for carregado, ative-o conforme mostrado na imagem:



Etapa 6. Na guia Resources, abra a configuração de interface para especificar o IP fixo que você deseja atribuir ao aplicativo, como mostrado na imagem:



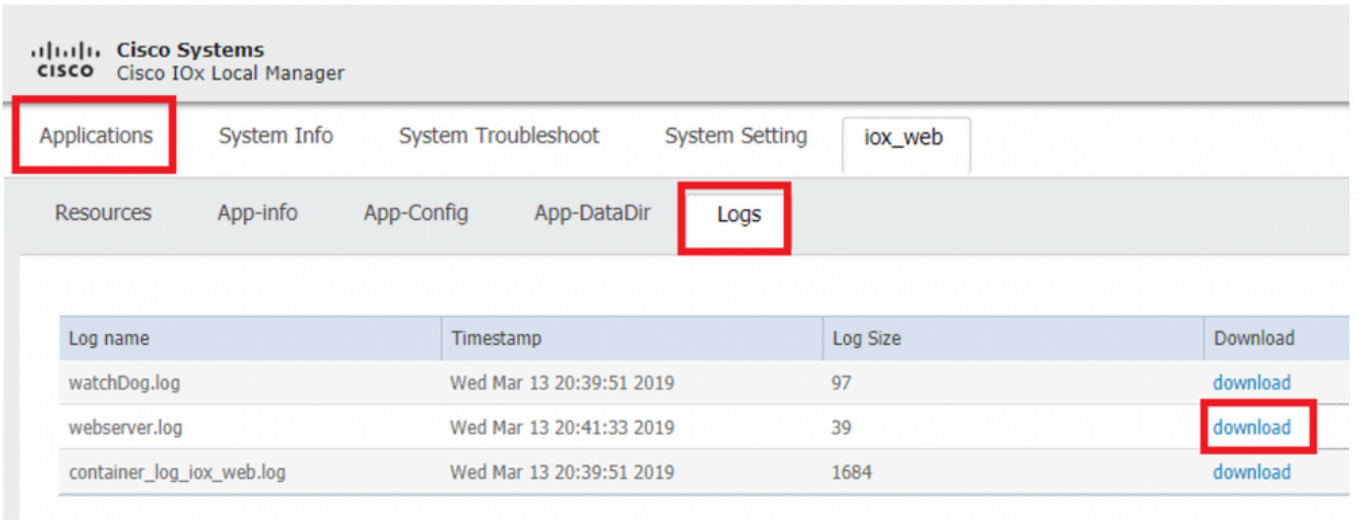
Passo 7. Clique em OK e em Ativar. Quando a ação for concluída, navegue de volta para a página principal do Gerenciador local (botão Aplicativos no menu superior) e inicie o aplicativo conforme mostrado na imagem:



Depois de passar por essas etapas, seu aplicativo estará pronto para ser executado.

Troubleshooting

Para resolver problemas de configuração, verifique o arquivo de log que você cria no script Python usando um gerenciador local. Navegue até Applications, clique em Manage no aplicativo iox_web e selecione a guia Logs como mostrado na imagem:



Sobre esta tradução
A Cisco traduziu este documento com a ajuda de tecnologias de tradução automática e humana para oferecer conteúdo de suporte aos seus usuários no seu próprio idioma, independentemente da localização.

Observe que mesmo a melhor tradução automática não será tão precisa quanto as realizadas por um tradutor profissional.

A Cisco Systems, Inc. não se responsabiliza pela precisão destas traduções e recomenda que o documento original em inglês (link fornecido) seja sempre consultado.