Creazione di applicazioni IOx con Vagrant e Virtualbox/VMWare

Sommario

Introduzione

In questo documento viene descritto come creare applicazioni IOx utilizzando Vagrant e Virtualbox e distribuirle nell'interfaccia grafica di IOx Local Manager.

Prerequisiti

Windows/ MAC Intel/ Linux

- Git
- Vagante
- Virtualbox

Basato su ARM MAC - M1/M2/M3

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

Per scaricare:

- Vagante
- VirtualBox

Procedura per impostare l'ambiente di creazione utilizzando

Vagrant

Riepilogo delle azioni

- La configurazione del file vagrante consente di configurare un ambiente VM in base all'architettura del computer host.
- Configura la VM per l'utilizzo di VMware Fusion o VirtualBox, a seconda dell'architettura
- Fornisce alla VM il software e gli strumenti necessari, tra cui QEMU (Quick EMUlator), Docker e ioxclient.
- La configurazione crea automaticamente un'applicazione iperf di esempio per i dispositivi della piattaforma Cisco di destinazione amd64.

Passaggio 1. Duplicare il repository Github nel sistema locale:

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

In alternativa, copiare e incollare il contenuto dell'enclosure di configurazione in "Vagrantfile". In questo modo viene creato un file con il nome "Vagrantfile" nel sistema locale:

```
# -*- 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
```

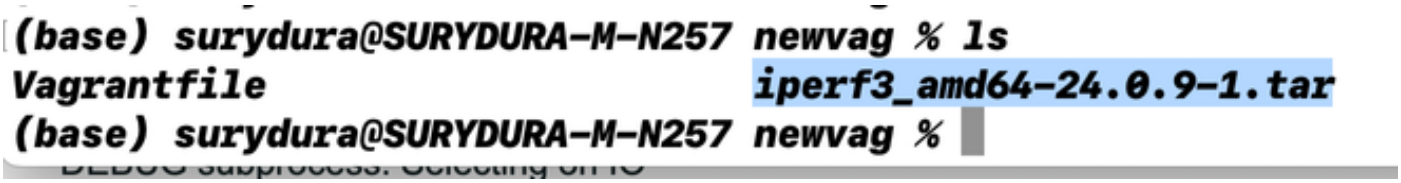
```
  SHELL
end
```

Passaggio 2. Assicurarsi che la riga "export VER="5:24.0.9-1~ubuntu.22.04~jammy" non contenga commenti e che tutte le altre istruzioni di esportazione contengano commenti. Corrisponde alla versione del Docker Engine che si desidera installare in questo 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"
```

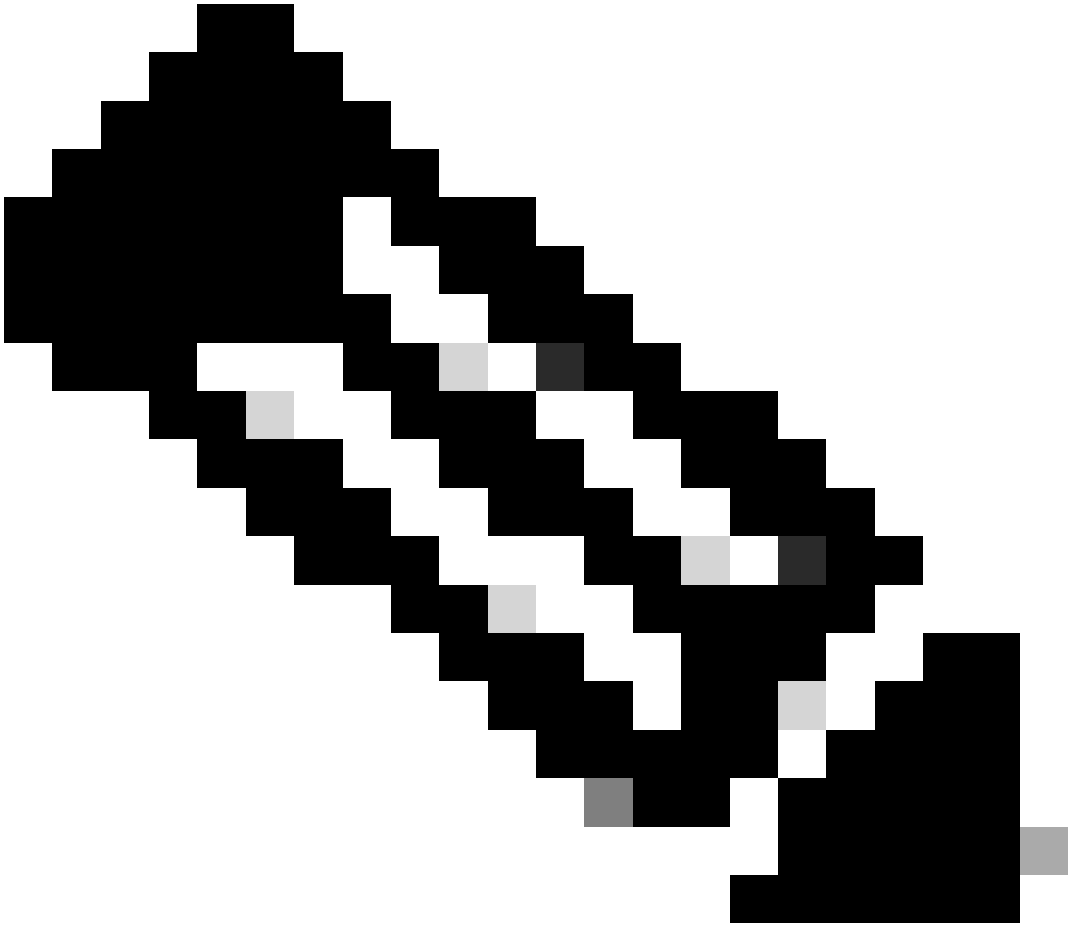
Passaggio 3. Avviare l'ambiente Vagrant con il comando vagrant up nella directory in cui si trova il file Vagrant e osservare che l'applicazione IOSx iperf per il file tar amd64 è stata generata correttamente:

```
vagrant up
```



Procedura per la creazione di un'applicazione IOx personalizzata

In questa sezione viene descritto come creare un'applicazione IOx personalizzata utilizzando l'ambiente vagrant.



> Nota: la directory "/vagrant" nella VM e la directory che contiene il "Vagrantfile" nel sistema host sono sincronizzate.

Come mostrato nell'immagine, il nuovo file.js viene creato all'interno della VM ed è accessibile anche sul 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 %
```

Passaggio 1. Clonare un'applicazione di esempio nella stessa cartella in cui si trova "Vagrantfile". Nell'esempio viene utilizzata l'applicazione "iox-multiarch-ninx-nyancat-sample":

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

Passaggio 2. SSH sul dispositivo vagante:

```
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:~$
```

Passaggio 3. Compilare l'applicazione:

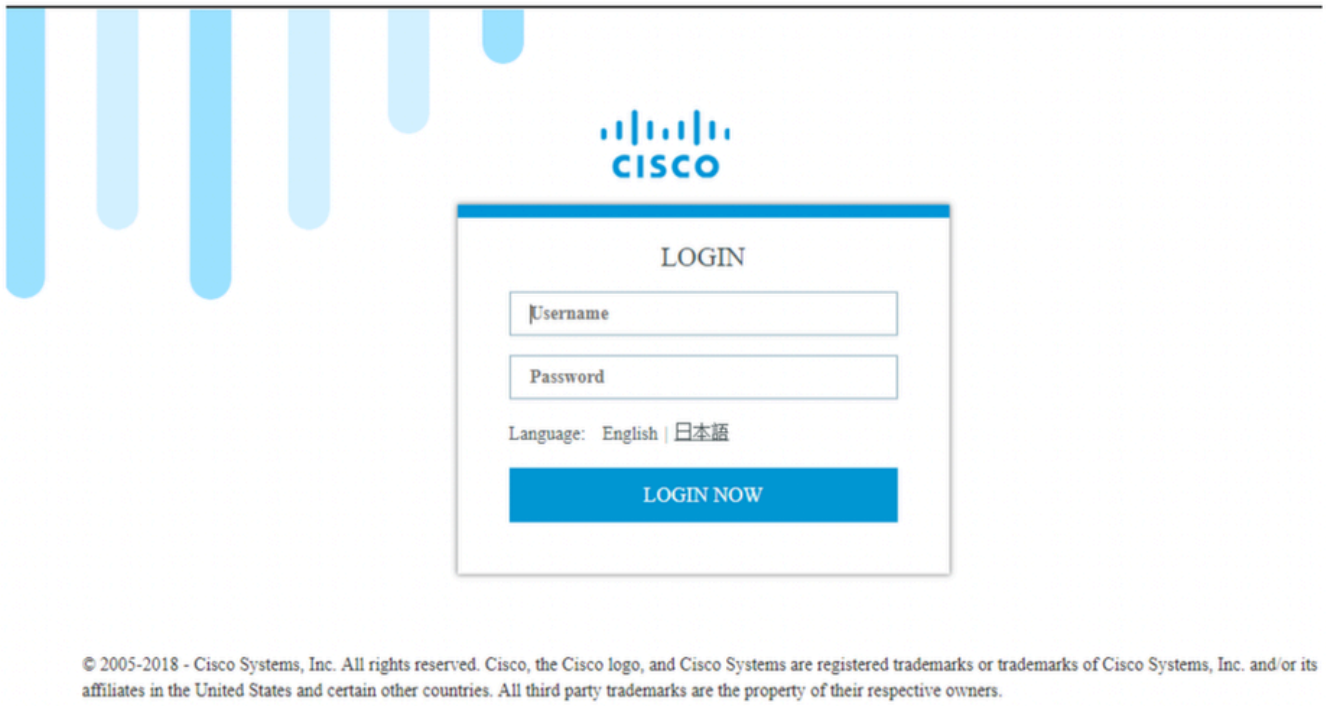
```
cd /vagrant/iox-multiarch-nginx-nyancat-sample/
chmod +x build
sh ./build
```

Al termine del processo di generazione, è possibile disporre di due applicazioni IOx pronte per l'installazione ("iox-amd64-nginx-nyancat-sample.tar.gz" per amd64 e "iox-arm64-nyancat-sample.tar.gz" per le piattaforme di destinazione):

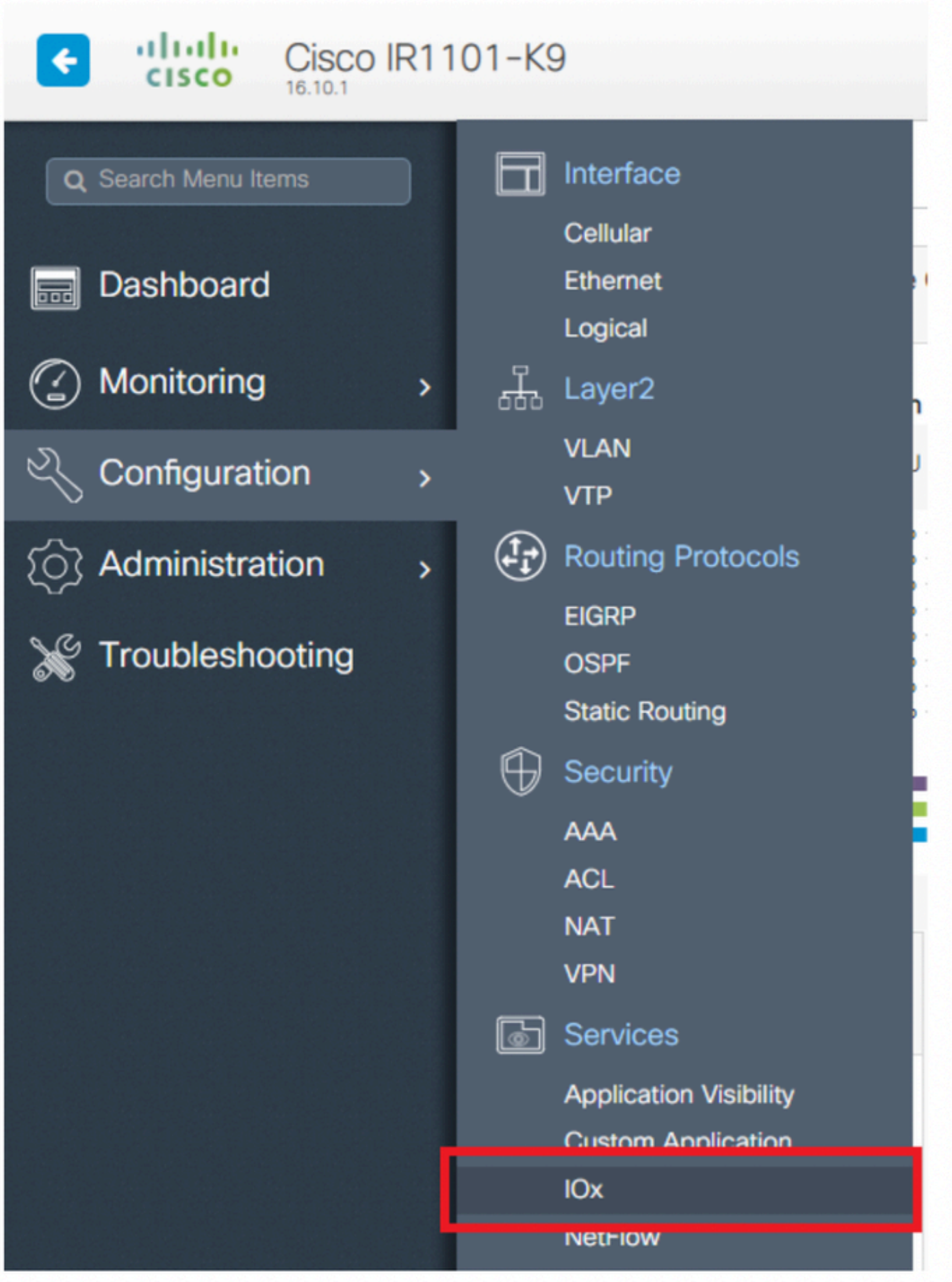
```
Package docker image iox-arm64-nginx-nyancat-sample at /vagrant/iox-multiarch-nginx-nyancat-sample/iox-arm64-nginx-nyancat-sample.tar.gz
vagrant@vagrant:/vagrant/iox-multiarch-nginx-nyancat-sample$ ls
Dockerfile  README.md  images                        iox-arm64-nginx-nyancat-sample.tar.gz  nyan-cat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

Distribuire l'applicazione IOx

Passaggio 1. Accedere a IR1101 tramite l'interfaccia Web:



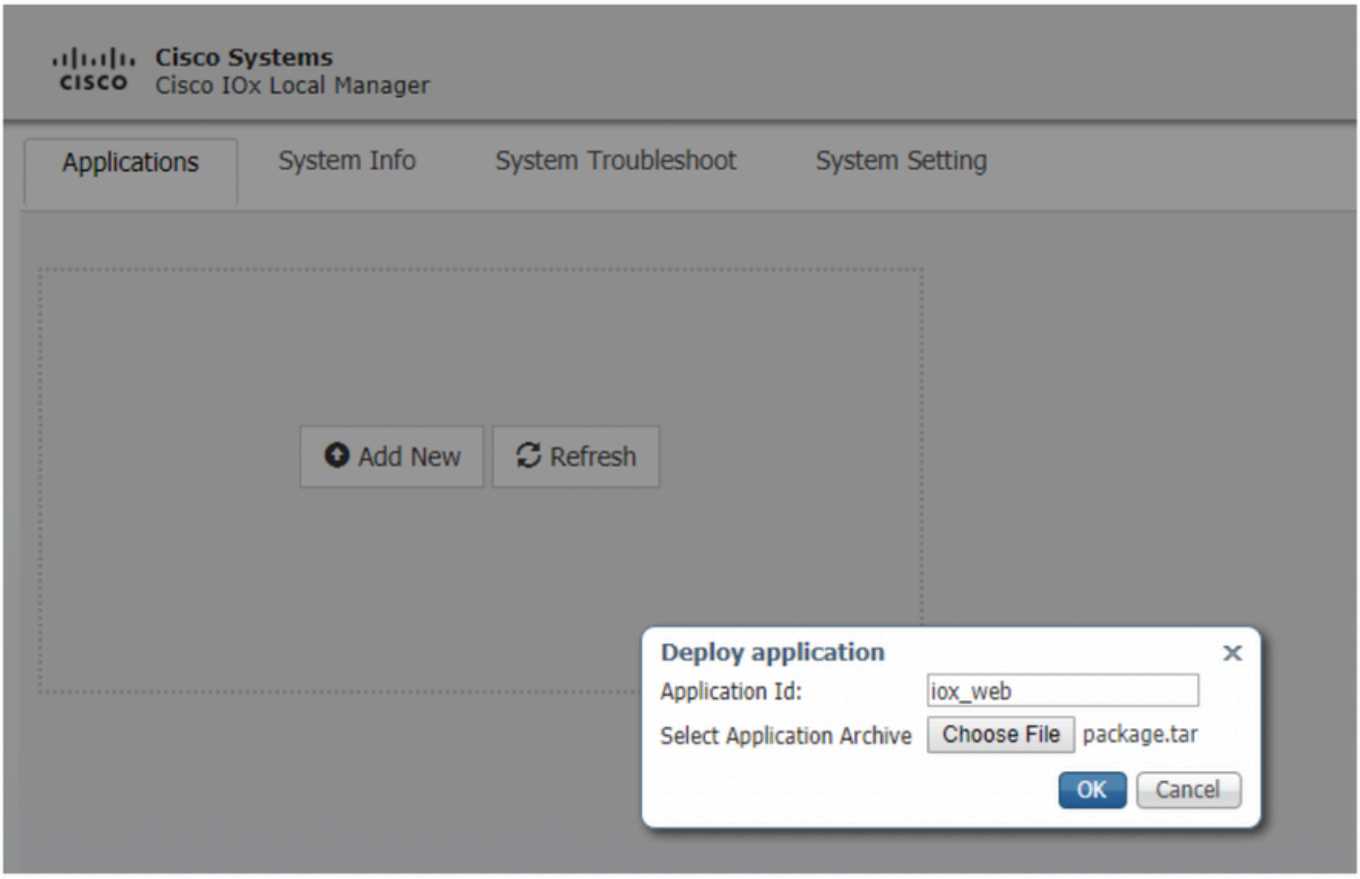
Passaggio 2. Utilizzare l'account con privilegi 15:



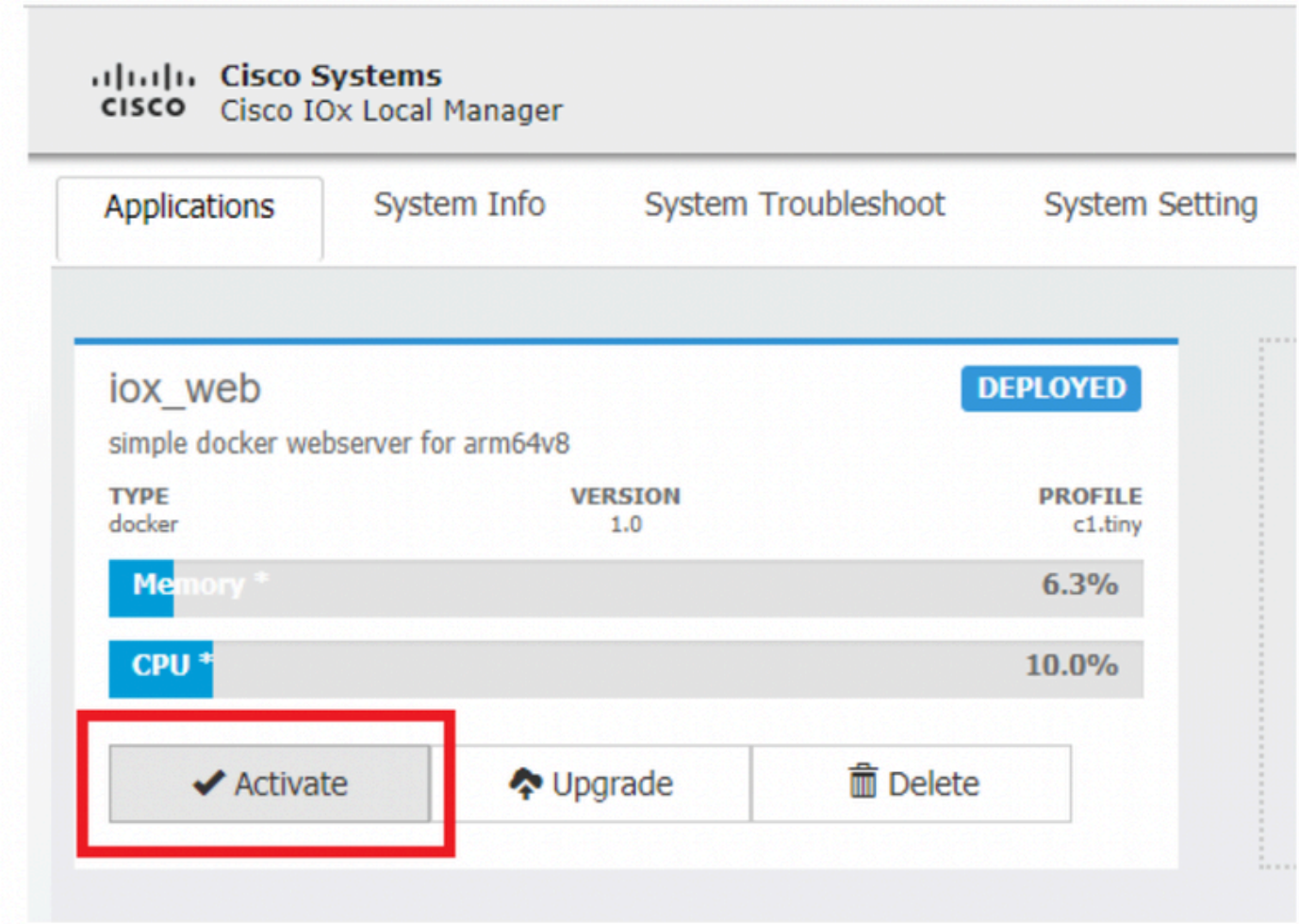
Passaggio 3. Nell'accesso a IOx Local Manager, utilizzare lo stesso account per continuare come

mostrato nell'immagine:

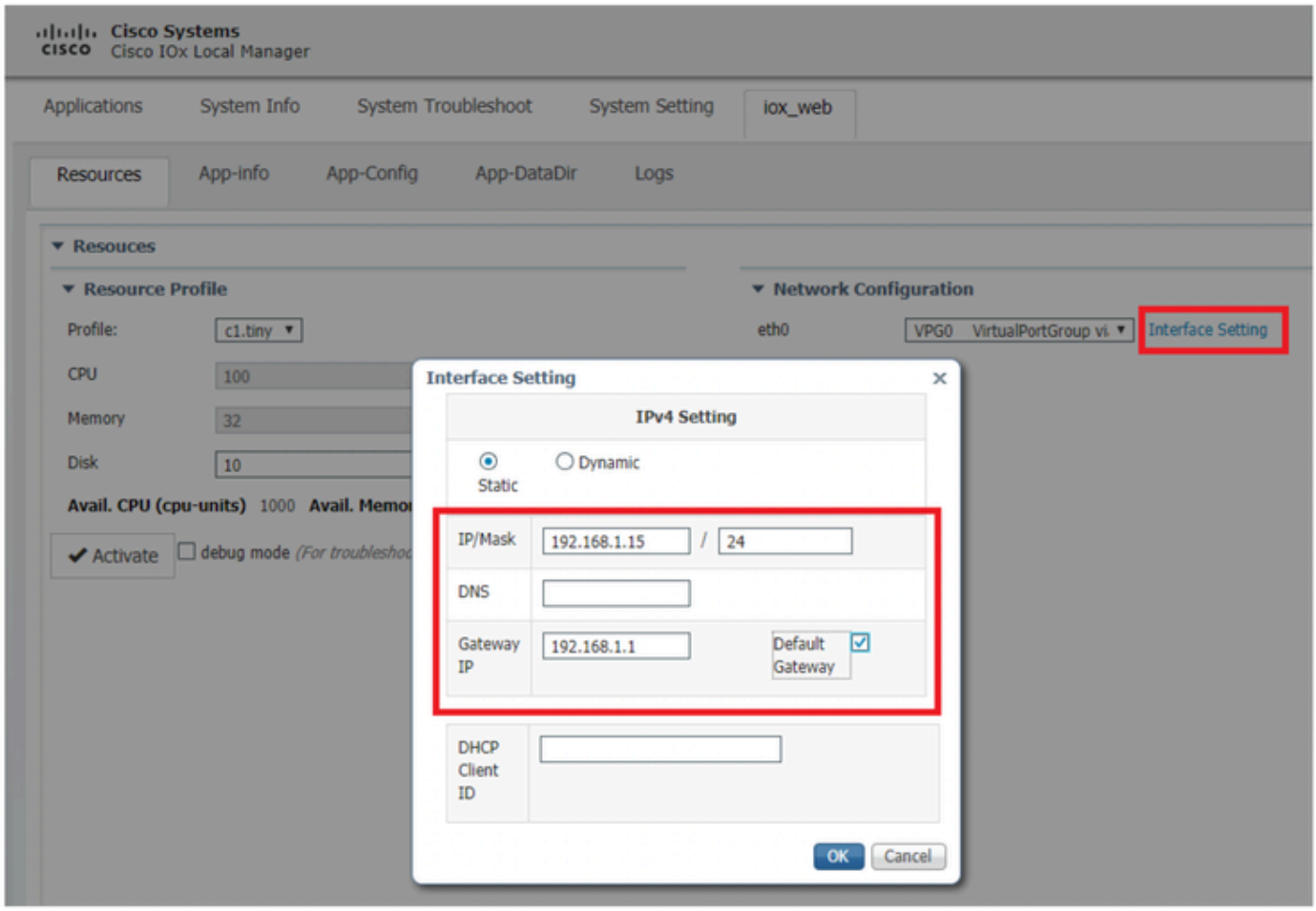
Passaggio 4. Fare clic su Add New (Aggiungi nuovo), selezionare un nome per l'applicazione IOx e scegliere il file package.tar creato nel passo 3 della sezione Procedure to Set Up Build Environment Using Vagrant (Procedura di configurazione dell'ambiente di generazione tramite Vagrant), come mostrato nell'immagine:



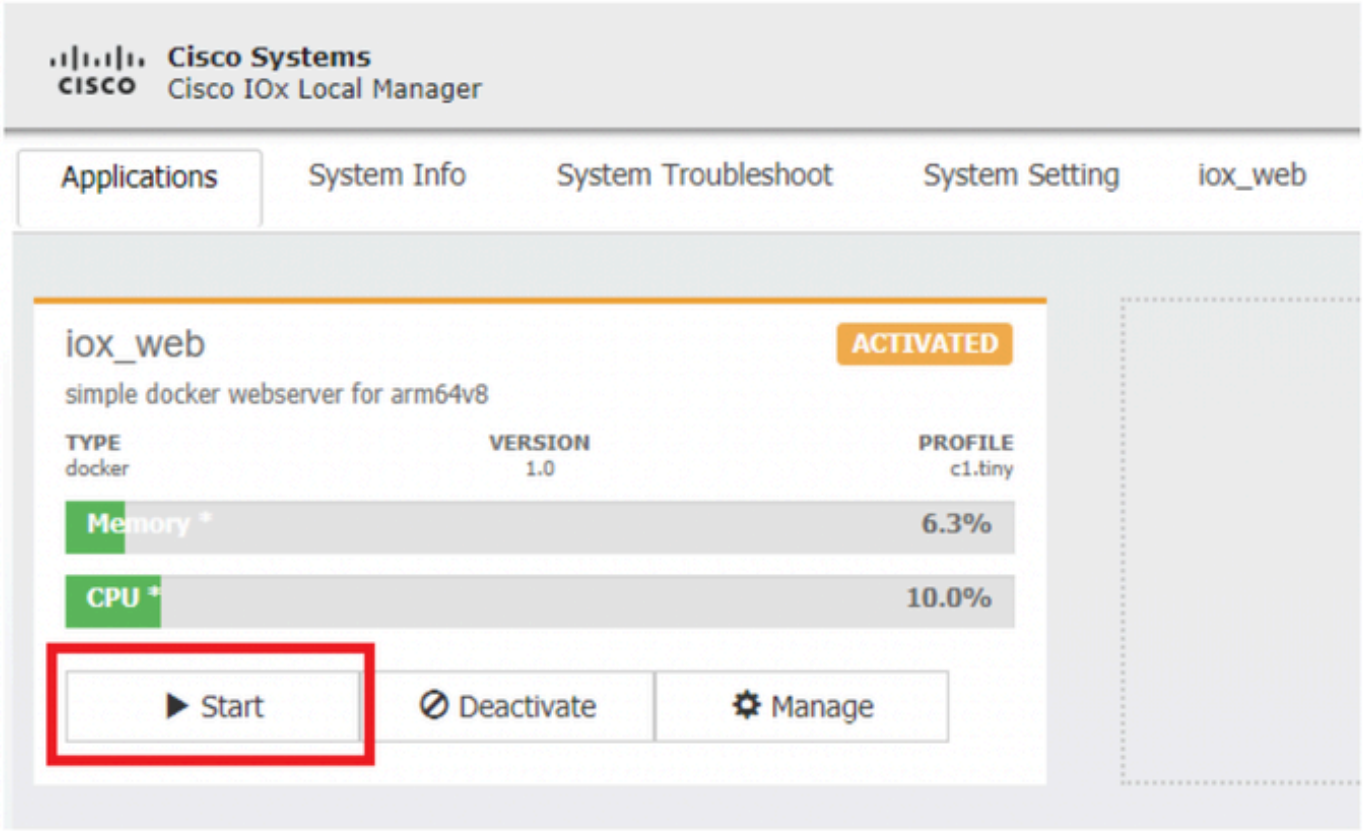
Passaggio 5. Una volta caricato il pacchetto, attivarlo come mostrato nell'immagine:



Passaggio 6. Nella scheda Resources, aprire l'impostazione dell'interfaccia per specificare l'indirizzo IP fisso che si desidera assegnare all'app, come mostrato nell'immagine:



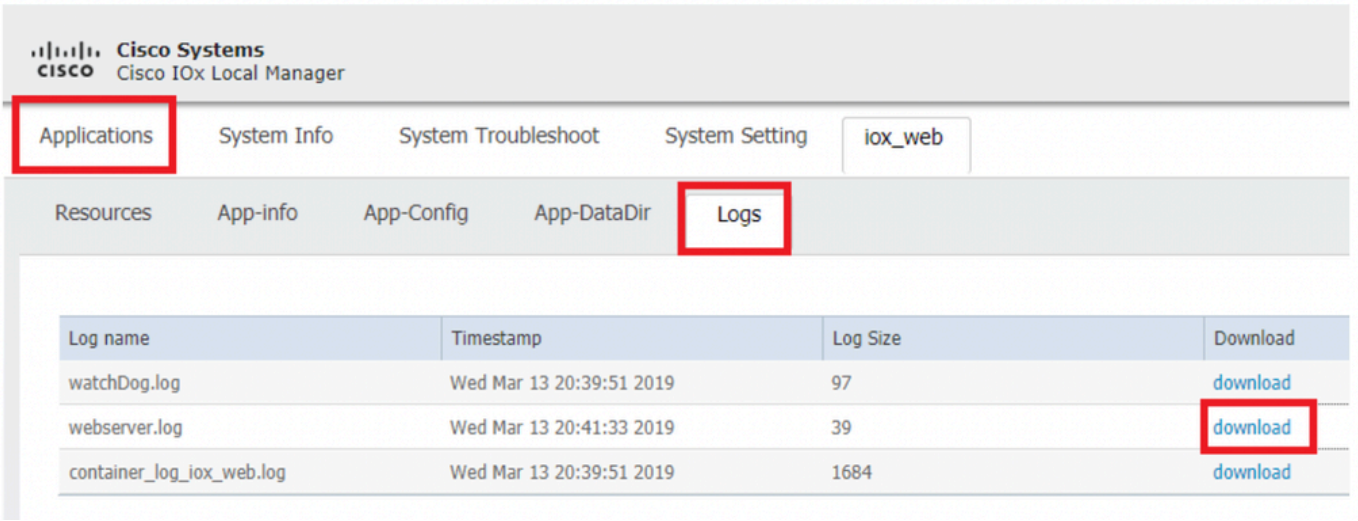
Passaggio 7. Fare clic su OK, quindi su Attiva. Una volta completata l'azione, tornare alla pagina principale di Gestione locale (pulsante Applicazioni nel menu superiore), quindi avviare l'applicazione come mostrato nell'immagine:



Dopo aver eseguito questi passaggi, l'applicazione è pronta per l'esecuzione.

Risoluzione dei problemi

Per risolvere i problemi relativi alla configurazione, controllare il file di log creato nello script Python utilizzando un programma di gestione locale. Passare ad Applicazioni, fare clic su Gestisci nell'applicazione iox_web, quindi selezionare la scheda Log come mostrato nell'immagine:



Informazioni su questa traduzione

Cisco ha tradotto questo documento utilizzando una combinazione di tecnologie automatiche e umane per offrire ai nostri utenti in tutto il mondo contenuti di supporto nella propria lingua. Si noti che anche la migliore traduzione automatica non sarà mai accurata come quella fornita da un traduttore professionista. Cisco Systems, Inc. non si assume alcuna responsabilità per l’accuratezza di queste traduzioni e consiglia di consultare sempre il documento originale in inglese (disponibile al link fornito).