Bouw IOx-apps met Vagrant en Virtualbox/VMWare

Inhoud

Inleiding

In dit document wordt beschreven hoe u IOx-toepassingen maakt met Vagrant en Virtualization Box en deze implementeert in IOx Local Manager GUI.

Voorwaarden

Windows/ MAC Intel/ Linux

- Git
- zwerfster
- Virtualbox

MAC ARM-gebaseerd - M1/M2/M3

- Git
- zwerfster
- VMWare Fusion
- vagrant-VMware-desktop plug-in

Downloaden:

- zwerfster
- VirtualBox

Procedure voor het instellen van een bouwomgeving met Vagrant

Samenvatting van de acties

- De vagrantfile configuratie vormt een VM-omgeving op basis van de architectuur van de host-machine.
- De VM wordt geconfigureerd om VMware Fusion of VirtualBox te gebruiken, afhankelijk van de architectuur
- Het voorziet de VM van de nodige software en instrumenten, waaronder QEMU (Quick EMUlator), Docker en ioxclient.
- De configuratie bouwt automatisch een voorbeeldtoepassing iperf voor amd64 doelCisco-platformapparaten.

Stap 1. De Github-opslagplaats in uw lokale systeem klonen:

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

U kunt ook de inhoud van de configuratiebehuizing kopiëren en plakken naar "Vagrantfile". Hier wordt een bestand gemaakt met de naam "Vagrantfile" in het lokale systeem:

```
# -*- 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
```

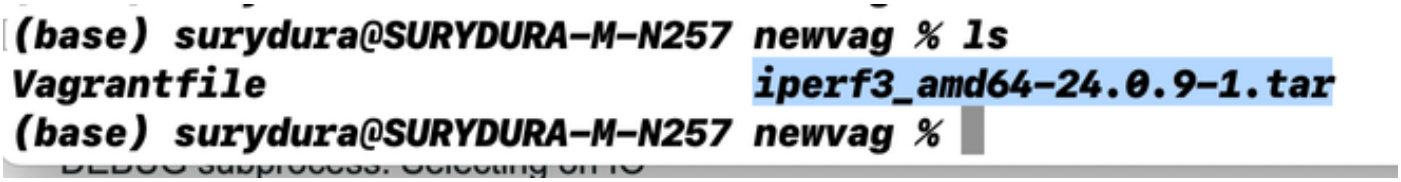
```
    #!/bin/bash
    # apt-cache madison docker-ce
    export VER="5:24.0.9-1~ubuntu.22.04~jammy"
    echo "!!! installing dependencies and packages !!!"
    apt-get update
    apt-get install -y ca-certificates curl unzip git pcregrep
    install -m 0755 -d /etc/apt/keyrings
    curl -fsSL https://download.docker.com/linux/ubuntu/gpg -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
```

Stap 2. Zorg ervoor dat de "export VER="5:24.0.9-1~ubuntu.22.04~jammy" lijn niet is gemarkeerd en dat alle andere exportoverzichten worden weergegeven. Dit komt overeen met de Docker Engine-versie die u in deze Vagrant-omgeving wilt installeren:

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

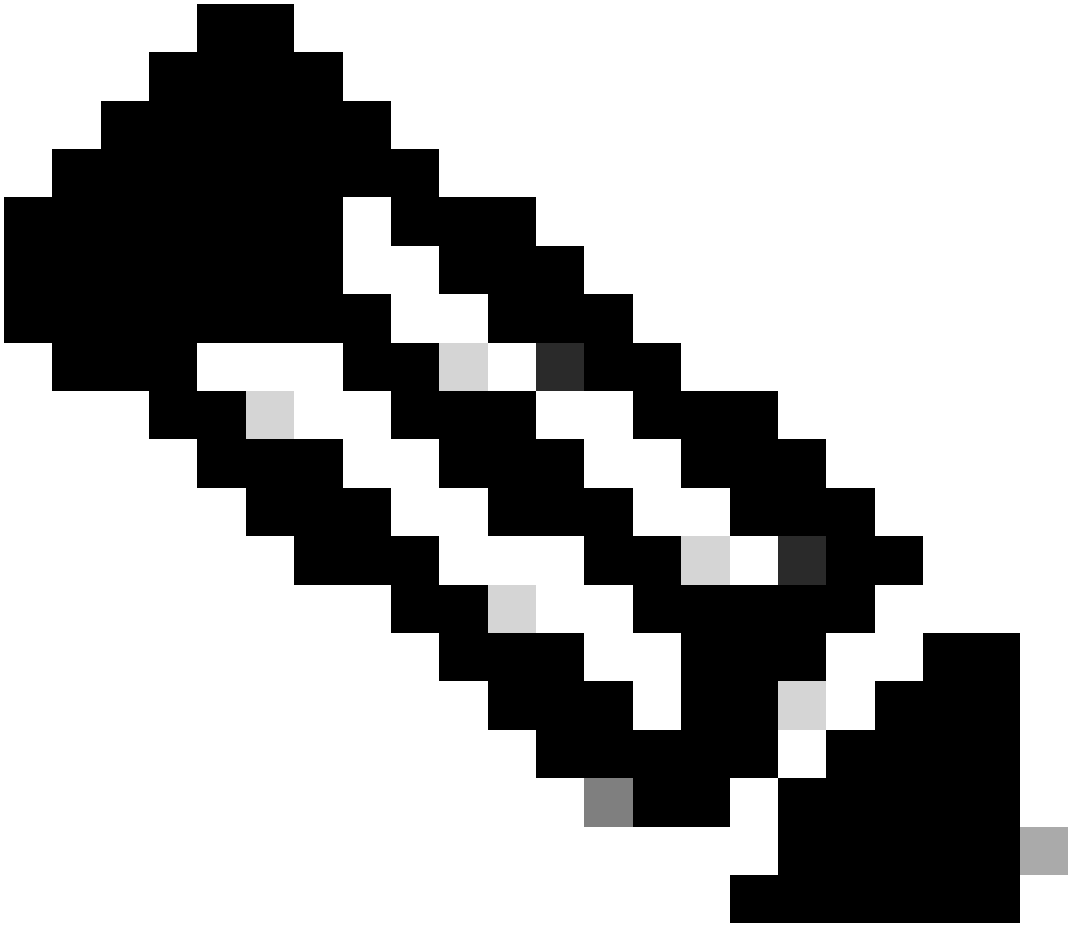
Stap 3. Start de Vagrantomgeving met de opdracht vagrant up in de map waarin het Vagrantbestand zich bevindt en bekijk een succesvolle opbouw van de iperf IOx-toepassing voor amd64 tar-bestand:

```
vagrant up
```



Procedure om een aangepaste IOSx-toepassing te bouwen

In dit deel wordt beschreven hoe een aangepaste IOx-toepassing kan worden gebouwd met behulp van de vagrantomgeving.



Opmerking: de map "/vagrant" in de VM en de map met het "Vagrantfile" in het hostsysteem zijn synchroon.

Zoals in de afbeelding wordt getoond, wordt het bestand new.js binnen de VM gemaakt en is het ook toegankelijk op het hostsysteem:

```
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 %
```

Stap 1. Kloon een voorbeeldtoepassing naar dezelfde map waarin "Vagrantfile" zich bevindt. In dit voorbeeld wordt "iox-multiarch-nginx-nyancat-sample" toegepast:

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

Stap 2. SSH in de schotelmachine:

```
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:~$
```

Stap 3. Bouw de applicatie:

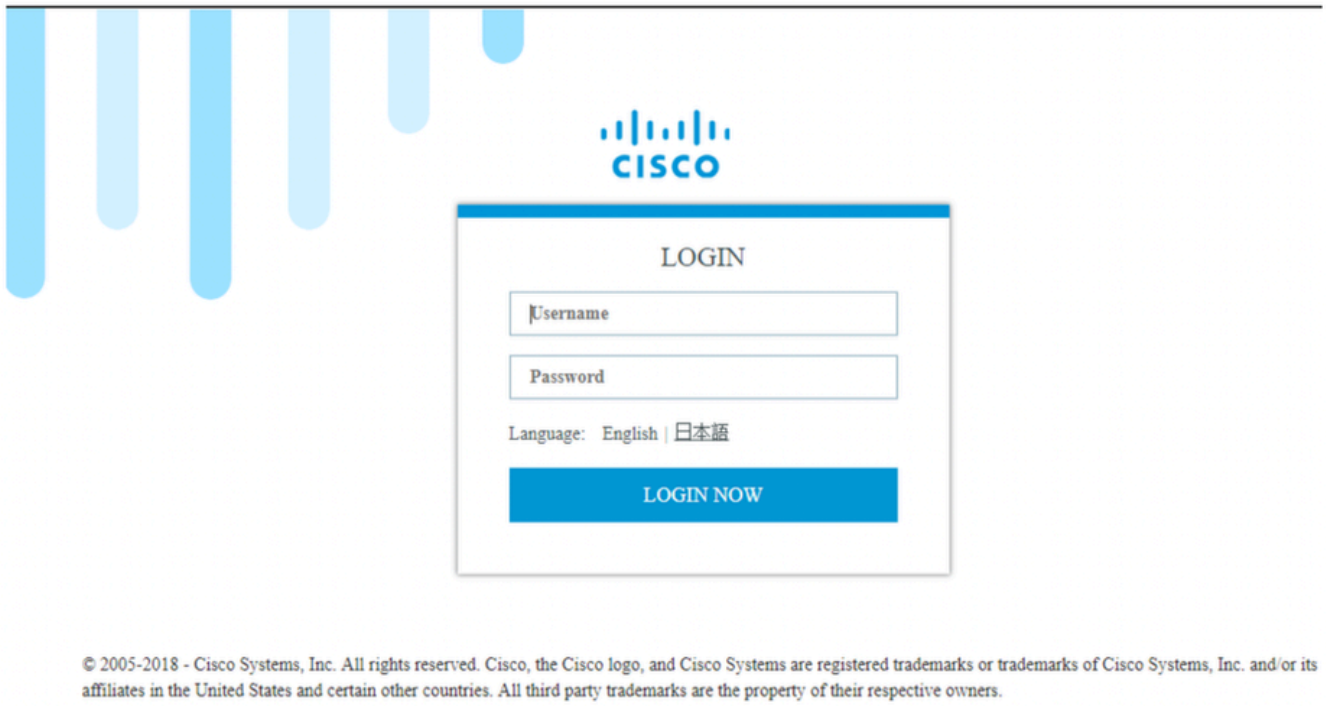
```
cd /vagrant/iox-multiarch-nginx-nyancat-sample/
chmod +x build
sh ./build
```

Nadat het bouwproces is voltooid, hebt u nu twee IOx-toepassingen klaar voor implementatie ("iox-amd64-nginx-nyancat-sample.tar.gz" voor amd64 en "iox-arm64-nginx-nyancat-sample.tar.gz" voor doelplatforms):

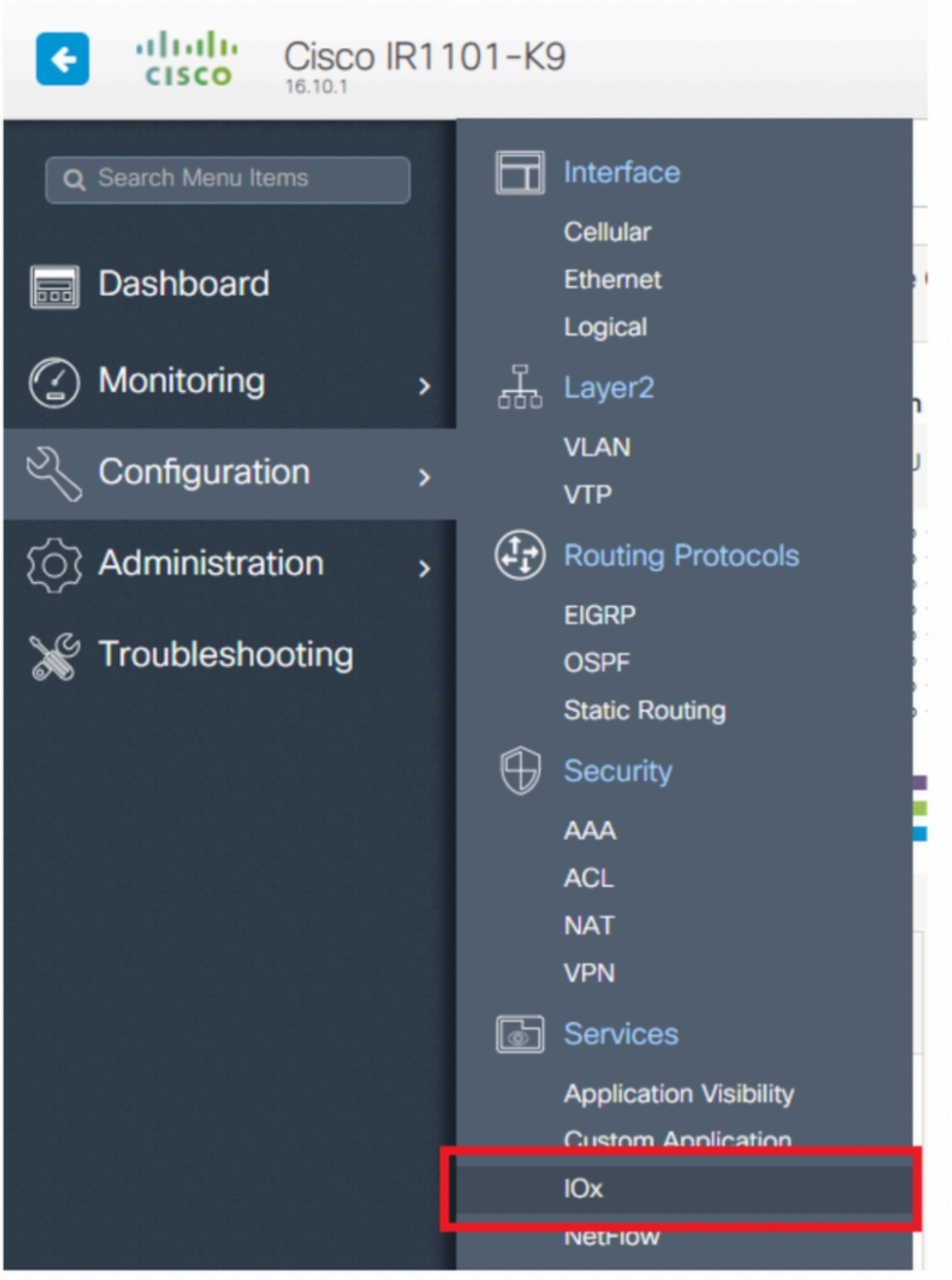
```
Package docker image iox-arm64-nginx-nyancat-sample at /vagrant/iox-multiarch-nginx-nyancat-sample/iox-arm64-nginx-nyancat-sample.tar.gz
vagrant@vagrant:/vagrant/iox-multiarch-nginx-nyancat-sample$ ls
Dockerfile  README.md  images                        iox-arm64-nginx-nyancat-sample.tar.gz  nyan-cat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

De IOSx-toepassing implementeren

Stap 1. Toegang tot de IR1101 met het gebruik van de webinterface:



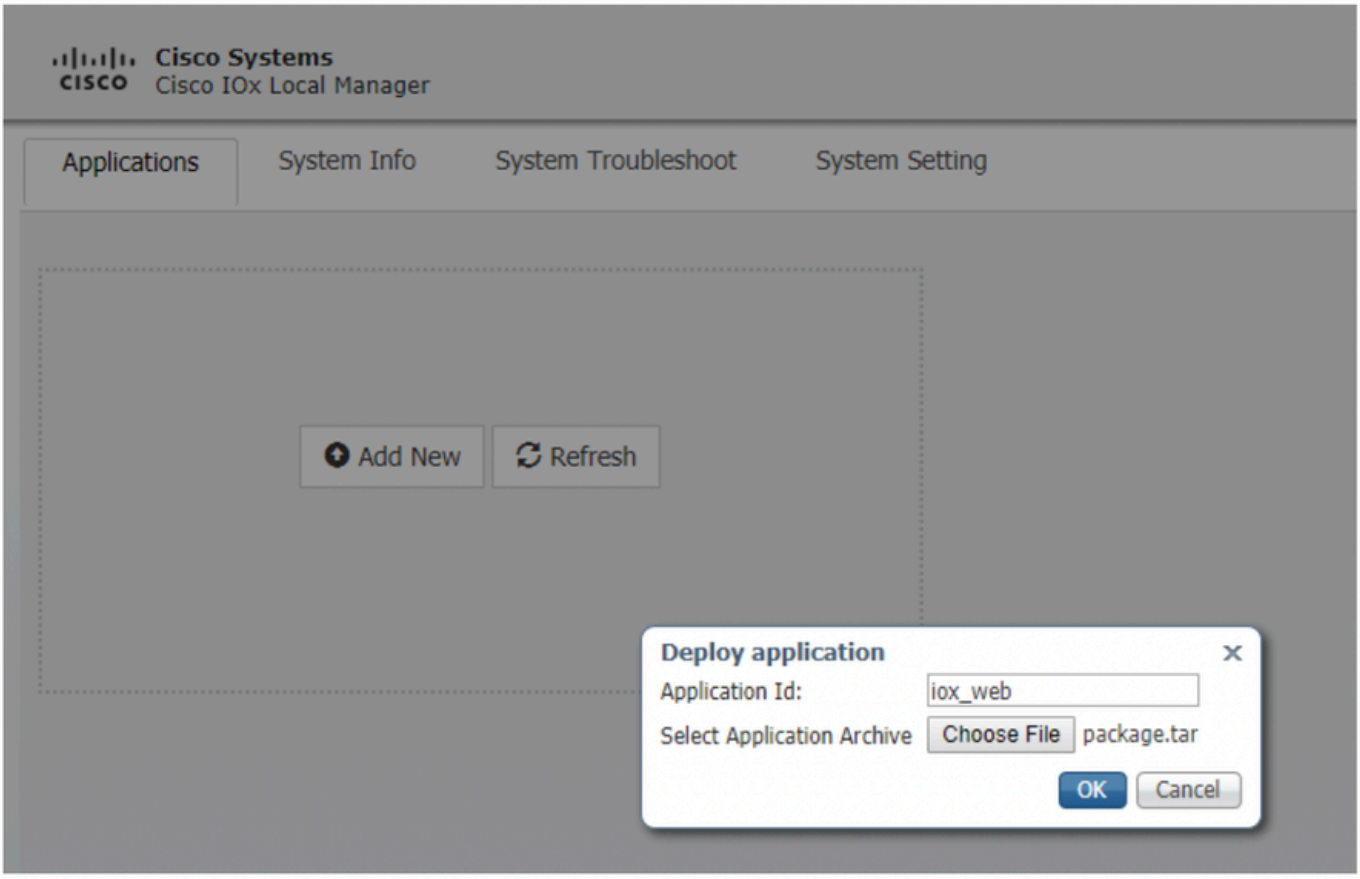
Stap 2. Gebruik de privilege 15-account:



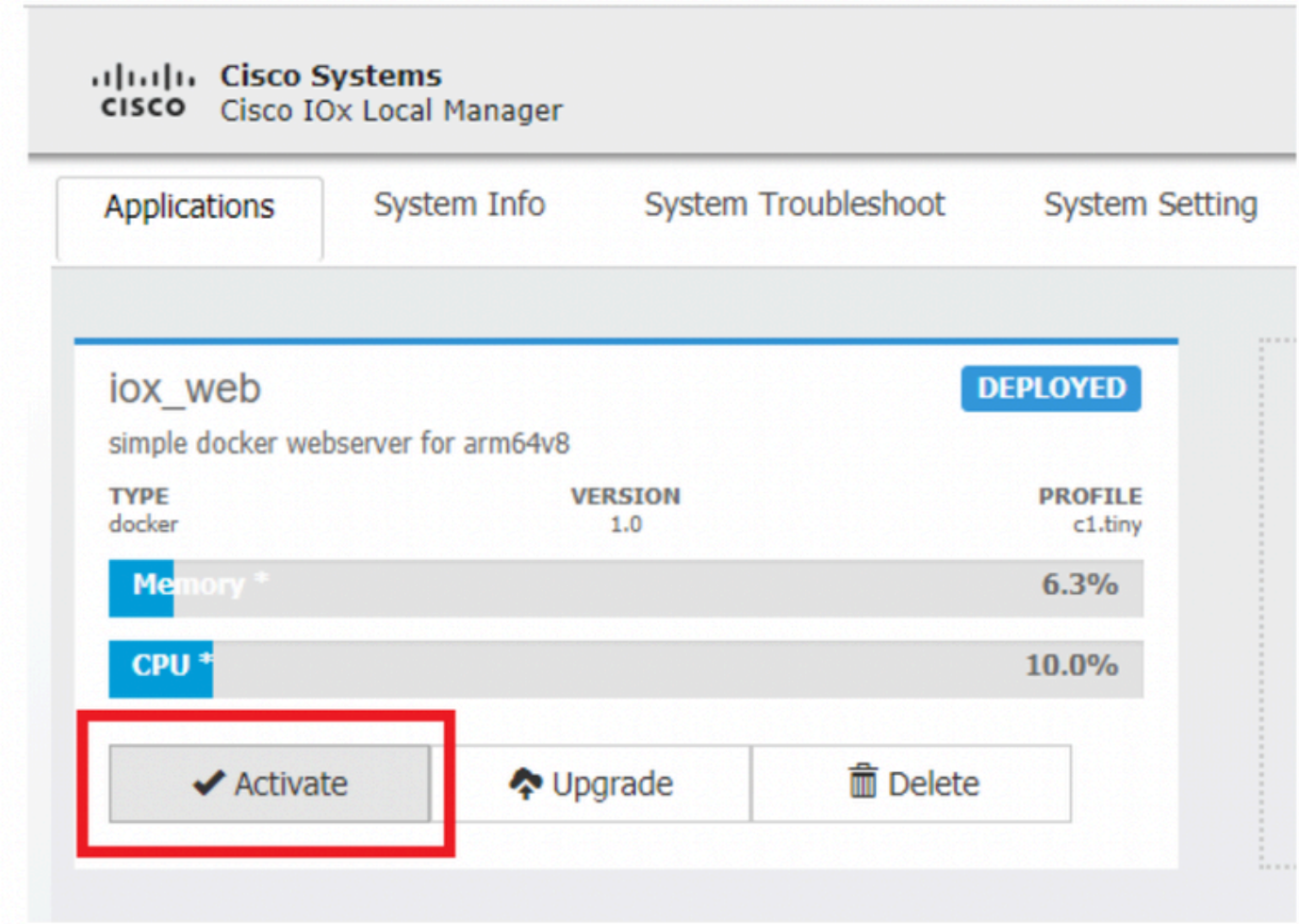
Stap 3. Gebruik in de inlognaam van IOx Local Manager dezelfde account om door te gaan zoals

in de afbeelding:

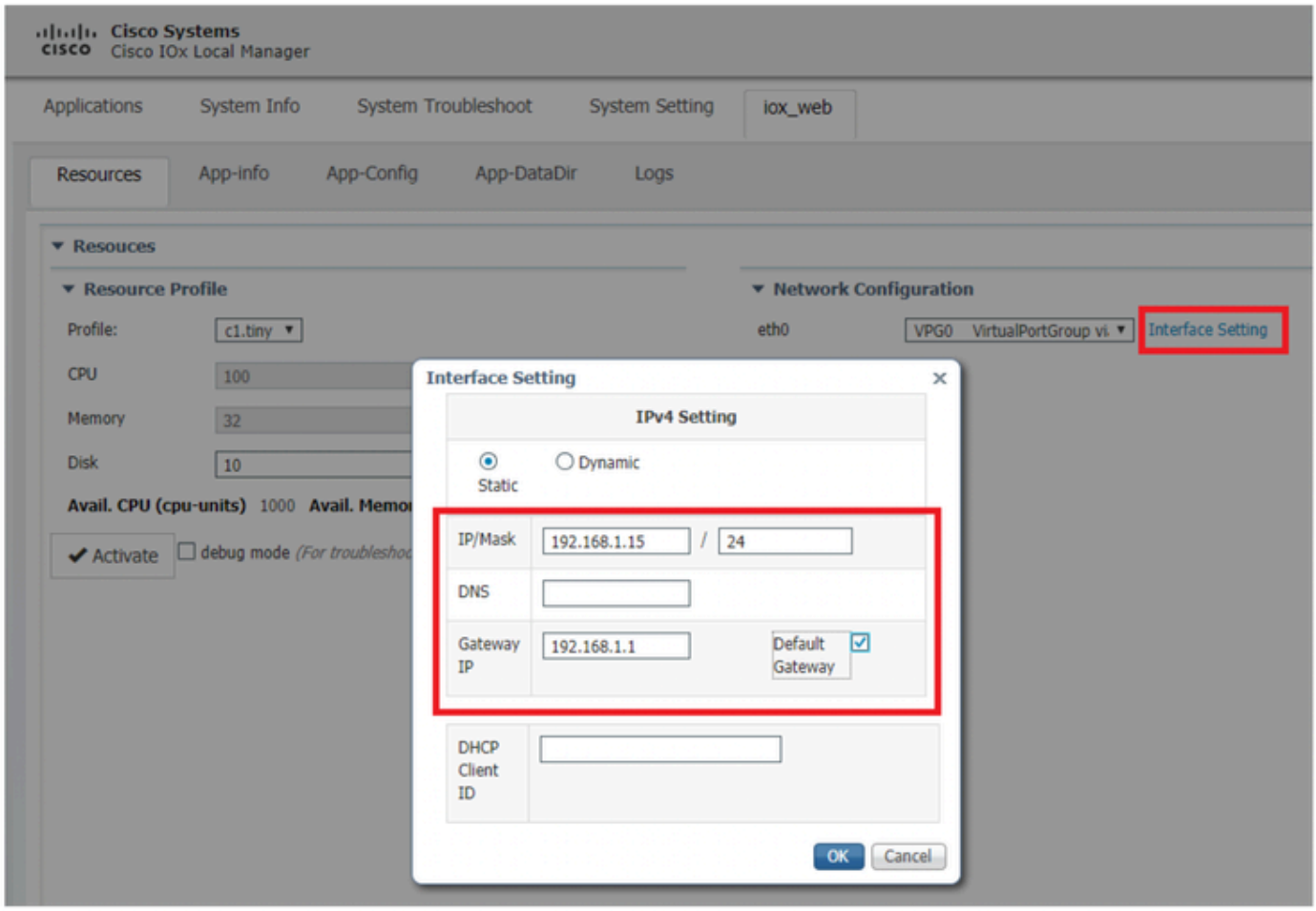
Stap 4. Klik op Add New, selecteer een naam voor de IOx-toepassing en kies het pakket.tar dat is gebouwd in stap 3 van de procedure voor het instellen van Build Environment met Vagrant sectie, zoals in de afbeelding:



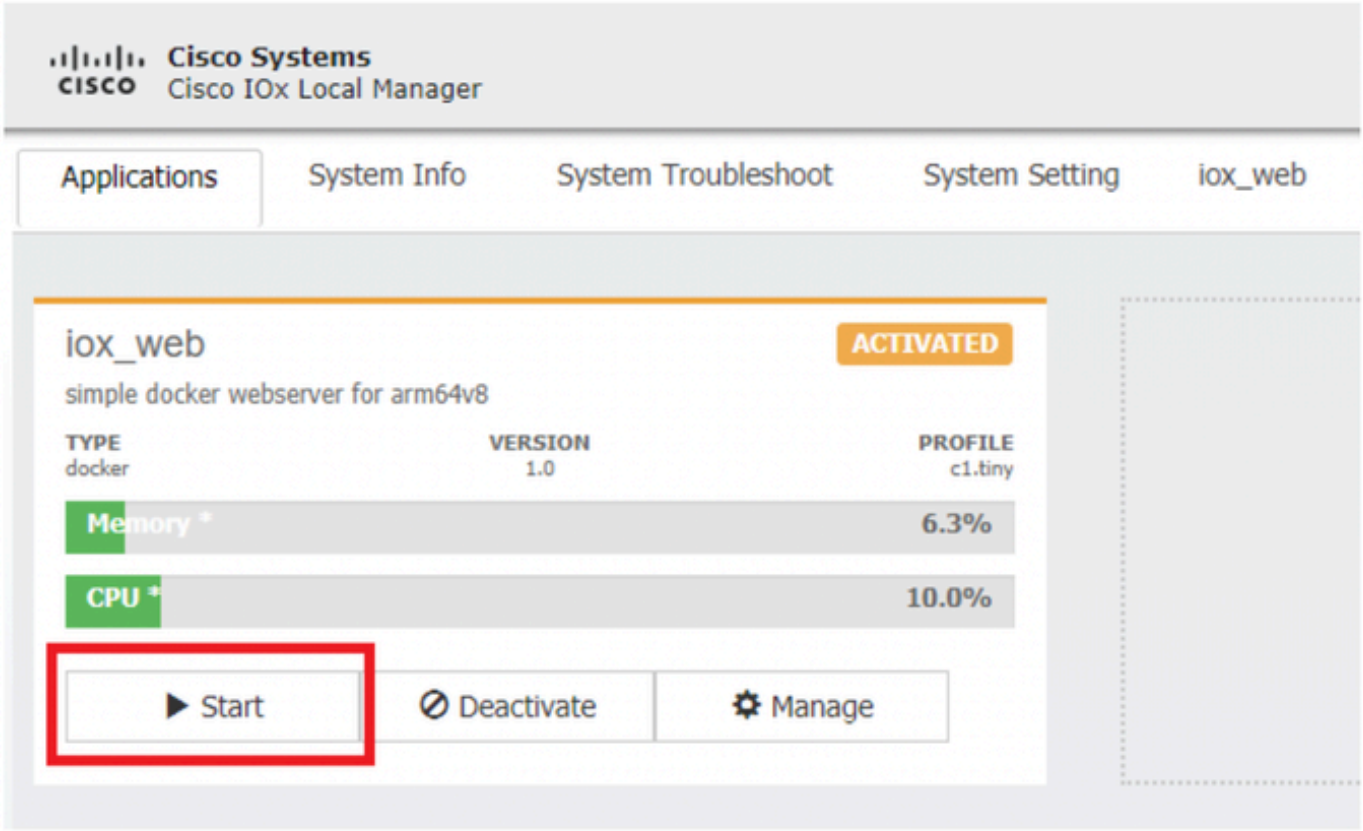
Stap 5. Nadat het pakket is geüpload, activeert u het zoals in de afbeelding:



Stap 6. In het tabblad Resources opent u de interface-instelling om het vaste IP te specificeren dat u wilt toewijzen aan de app zoals in de afbeelding:



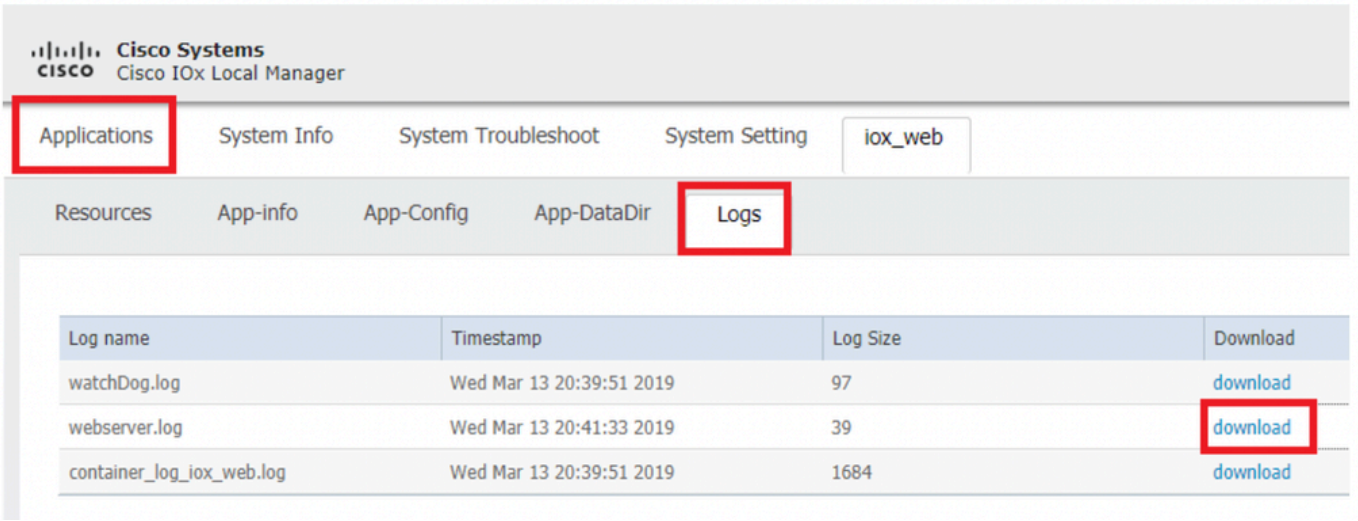
Stap 7. Klik op OK en vervolgens op Activeren. Nadat de actie is voltooid, gaat u terug naar de pagina Local Manager (Toepassingen in het bovenste menu) en start u de toepassing zoals in de afbeelding:



Nadat u deze stappen hebt doorlopen, is uw applicatie klaar om te worden uitgevoerd.

Problemen oplossen

Om uw configuratie problemen op te lossen, controleert u het logbestand dat u in het Python-script maakt met een lokale beheerder. Navigeer naar Toepassingen, klik op Beheer in de iox_web applicatie en selecteer vervolgens het tabblad Logs zoals in de afbeelding:



Over deze vertaling

Cisco heeft dit document vertaald via een combinatie van machine- en menselijke technologie om onze gebruikers wereldwijd ondersteuningscontent te bieden in hun eigen taal. Houd er rekening mee dat zelfs de beste machinevertaling niet net zo nauwkeurig is als die van een professionele vertaler. Cisco Systems, Inc. is niet aansprakelijk voor de nauwkeurigheid van deze vertalingen en raadt aan altijd het oorspronkelijke Engelstalige document (link) te raadplegen.