Erstellung von IOx-Anwendungen mit Vagrant und VirtualBox/VMWare

Inhalt

Einleitung

In diesem Dokument wird beschrieben, wie Sie IOx-Anwendungen mithilfe von Vagrant und VirtualBox erstellen und in der lokalen IOx-Manager-GUI bereitstellen.

Voraussetzungen

Windows/ MAC Intel/ Linux

- Git
- Vagrant
- VirtuelleBox

MAC-ARM-basiert - M1/M2/M3

- Git
- Vagrant
- VMware-Fusion
- vagrant-vmware-desktop-Plugin

Herunterladen:

- Vagrant
- VirtuelleBox

Verfahren zum Einrichten einer Buildumgebung mit Vagrant

Zusammenfassung der Maßnahmen

- Die vagrantfile-Konfiguration richtet eine VM-Umgebung basierend auf der Host-Rechnerarchitektur ein.
- Je nach Architektur wird das virtuelle System so konfiguriert, dass es entweder VMware Fusion oder VirtualBox verwendet
- Es stellt dem VM die notwendige Software und Tools zur Verfügung, darunter QEMU (Quick EMUlator) , Docker und ioxclient.
- Bei der Konfiguration wird automatisch eine iperf-Beispielanwendung für AMD64-Zielgeräte der Cisco Plattform erstellt.

Schritt 1: Klonen Sie das Github-Repository in Ihrem lokalen System:

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

Alternativ können Sie den Inhalt des Konfigurationsgehäuses kopieren und in "Vagrantfile" einfügen. Dadurch wird eine Datei mit dem Namen "Vagrantfile" im lokalen System erstellt:

```
# -*- 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
```

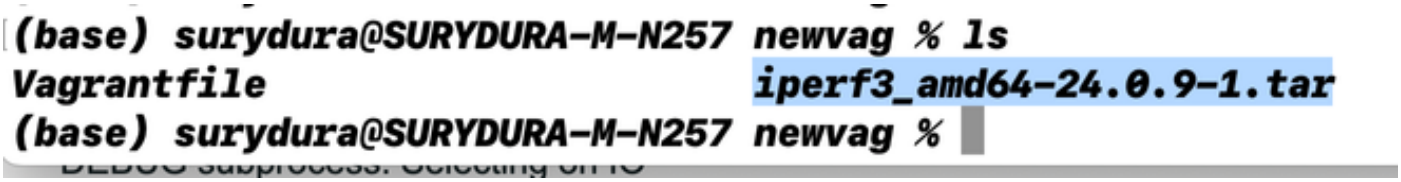
```
    #!/bin/bash
    # apt-cache madison docker-ce
    export VER="5:24.0.9-1~ubuntu.22.04~jammy"
    echo "!!! installing dependencies and packages !!!"
    apt-get update
    apt-get install -y ca-certificates curl unzip git pcregrep
    install -m 0755 -d /etc/apt/keyrings
    curl -fsSL https://download.docker.com/linux/ubuntu/gpg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

Schritt 2: Stellen Sie sicher, dass die Zeile "export VER="5:24.0.9-1~ubuntu.22.04~jammy" nicht kommentiert und alle anderen Exportanweisungen kommentiert sind. Dies entspricht der Docker-Engine-Version, die Sie in dieser Vagrant-Umgebung installieren möchten:

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

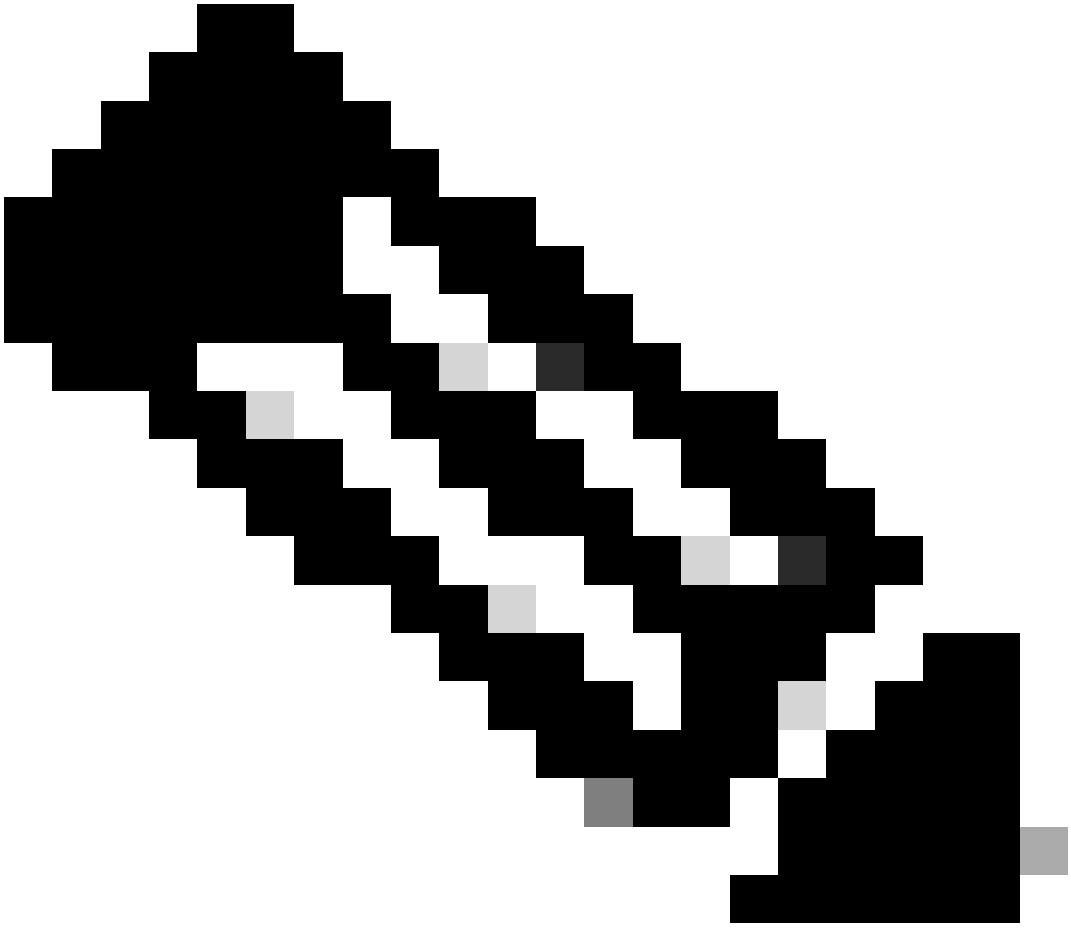
Schritt 3: Starten Sie die Vagrant-Umgebung mit dem Befehl vagrant up in dem Verzeichnis, in dem sich die Vagrantdatei befindet, und beobachten Sie eine erfolgreiche Erstellung der IOx-Anwendung iperf für amd64 tar-Datei:

```
vagrant up
```



Verfahren zum Erstellen einer benutzerdefinierten IOx-Anwendung

In diesem Abschnitt wird beschrieben, wie Sie eine benutzerdefinierte IOx-Anwendung mithilfe der vaganten Umgebung erstellen.



> Hinweis: Das Verzeichnis "/vagrant" im VM und das Verzeichnis, das die "Vagrantdatei" im Host-System enthält, sind synchronisiert.

Wie im Bild gezeigt, wird die Datei new.js innerhalb des virtuellen Systems erstellt und ist auch auf dem Host-System zugänglich:

```
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 %
```

Schritt 1: Klonen Sie eine Beispielanwendung in denselben Ordner, in dem sich "Vagrantfile" befindet. In diesem Beispiel wird die Anwendung "iox-multiarch-nginx-nyancat-sample" verwendet:

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

Schritt 2: SSH in die vagante Maschine:

```
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:~$
```

Schritt 3: Erstellen Sie die Anwendung:

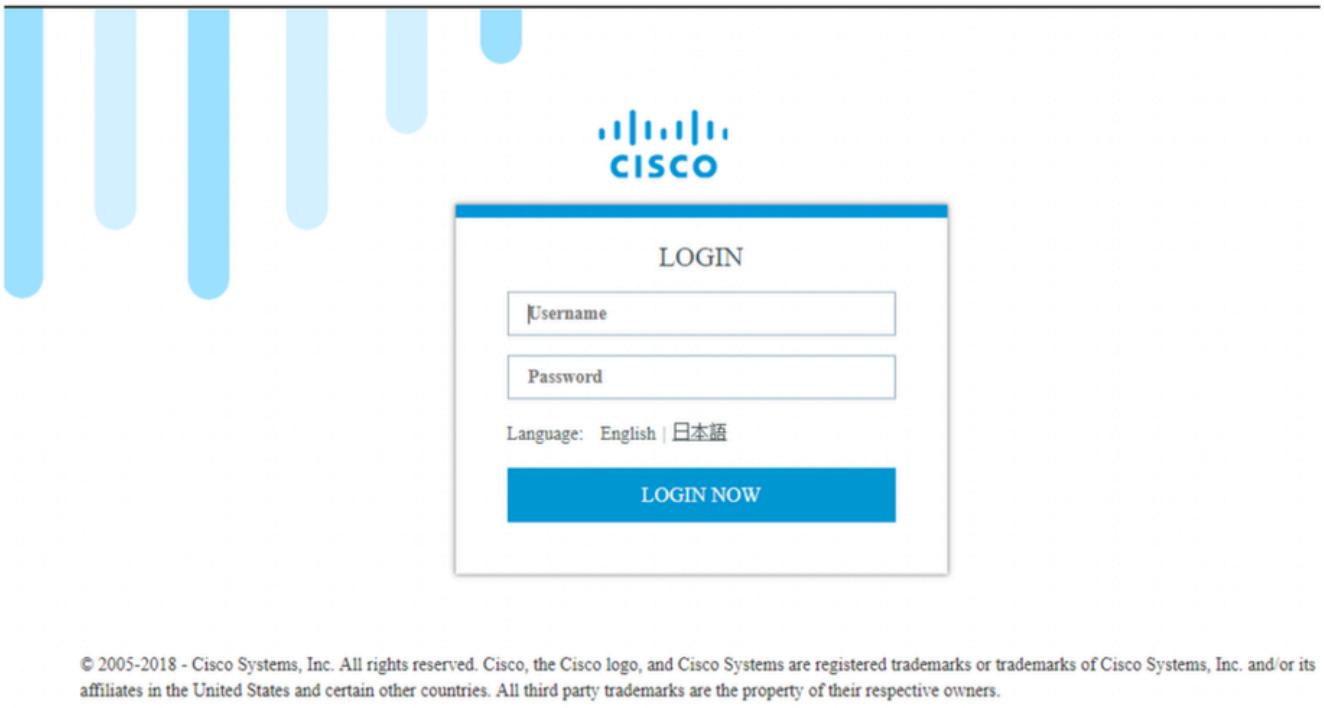
```
cd /vagrant/iox-multiarch-nginx-nyancat-sample/
chmod +x build
sh ./build
```

Nach Abschluss des Buildprozesses stehen Ihnen nun zwei IOx-Anwendungen zur Bereitstellung zur Verfügung ("iox-amd64-nginx-nyancat-sample.tar.gz" für amd64 und "iox-arm64-nginx-nyancat-sample.tar.gz" für Zielplattformen):

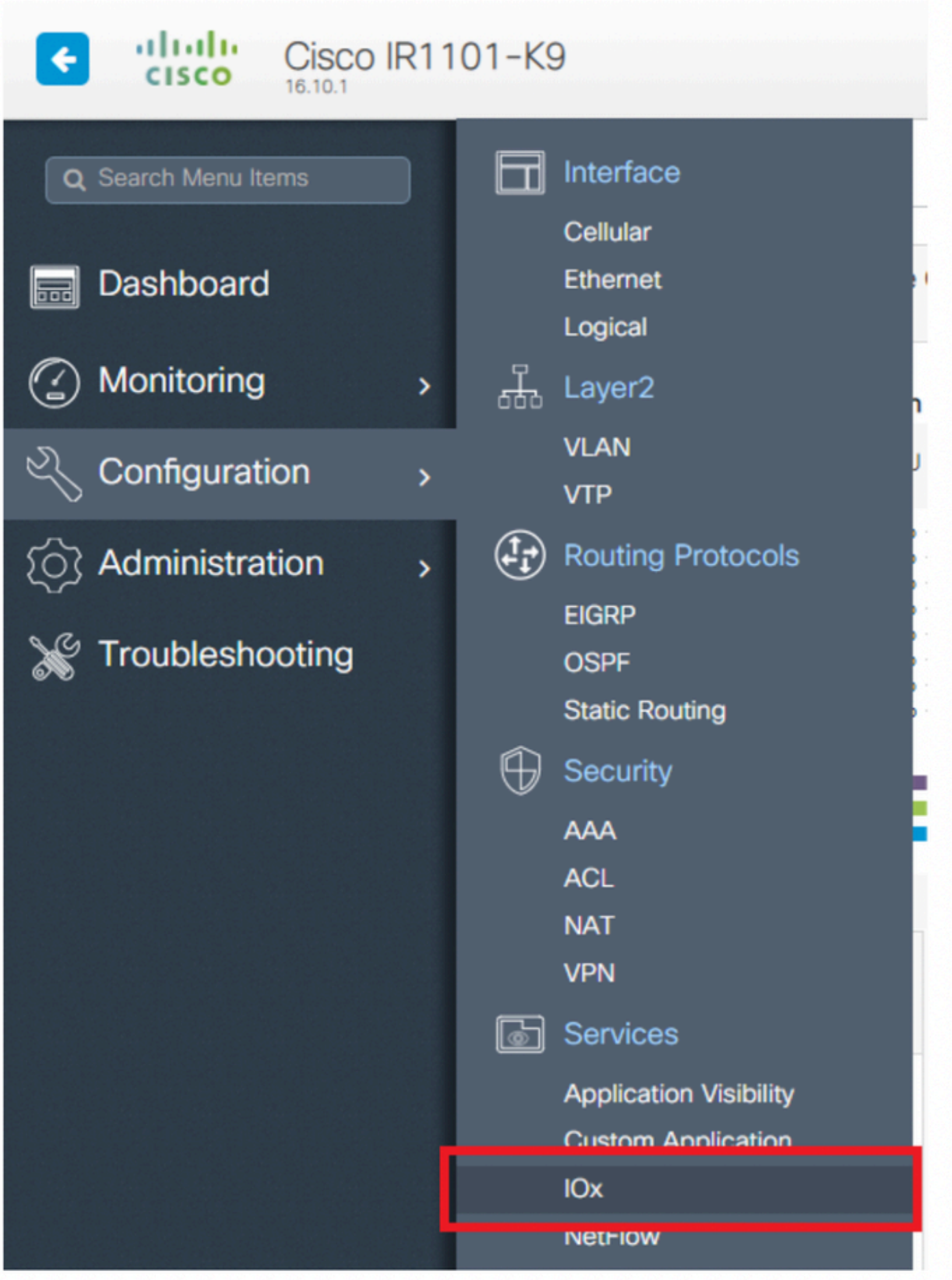
```
Package docker image iox-arm64-nginx-nyancat-sample at /vagrant/iox-multiarch-nginx-nyancat-sample/iox-arm64-nginx-nyancat-sample.tar.gz
vagrant@vagrant:/vagrant/iox-multiarch-nginx-nyancat-sample$ ls
Dockerfile  README.md  images                        iox-arm64-nginx-nyancat-sample.tar.gz  nyan-cat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

Bereitstellung der IOx-Anwendung

Schritt 1: Über die Webschnittstelle können Sie auf den IR1101 zugreifen:



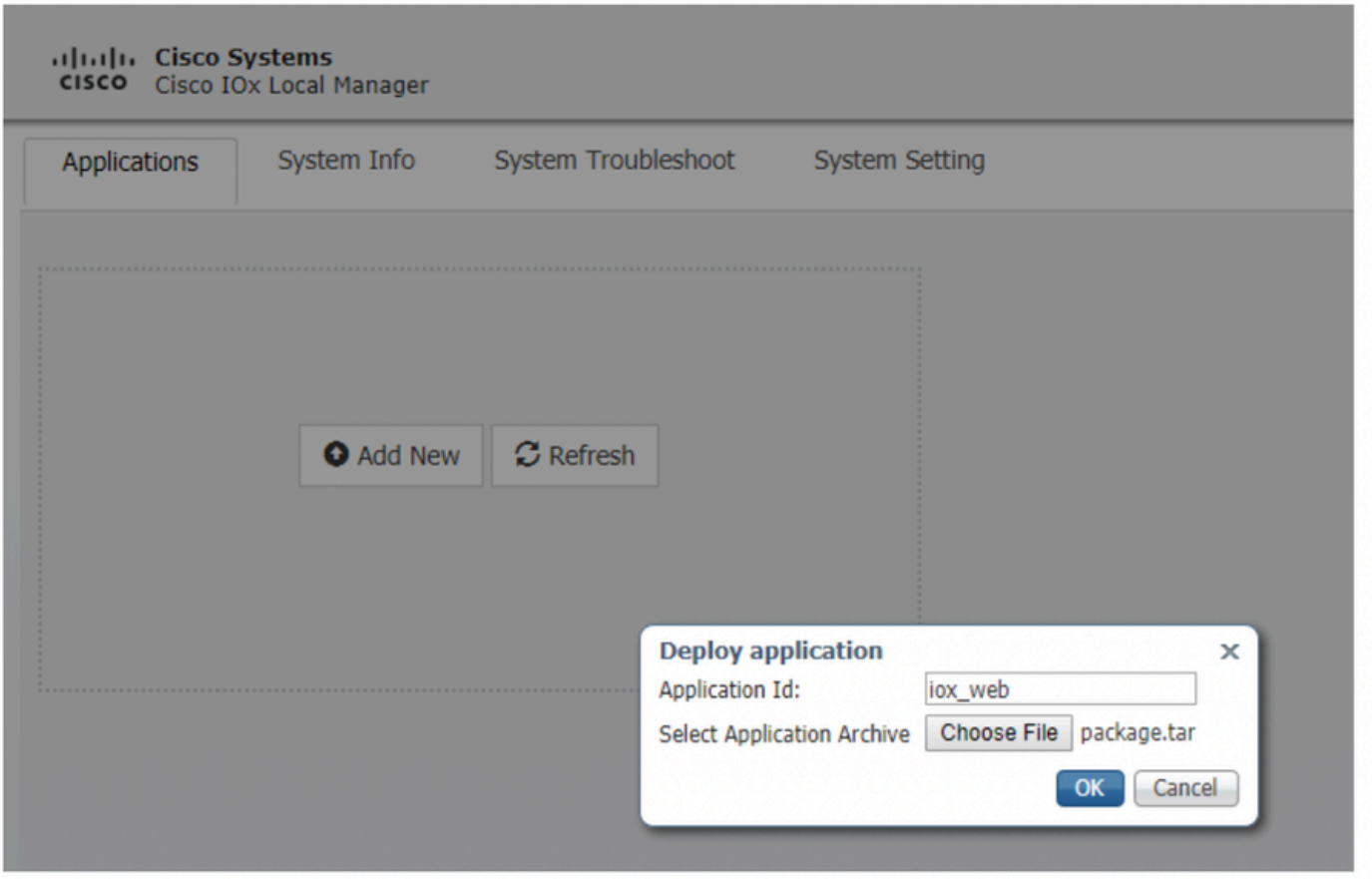
Schritt 2: Verwenden Sie das Konto mit der Berechtigung 15:



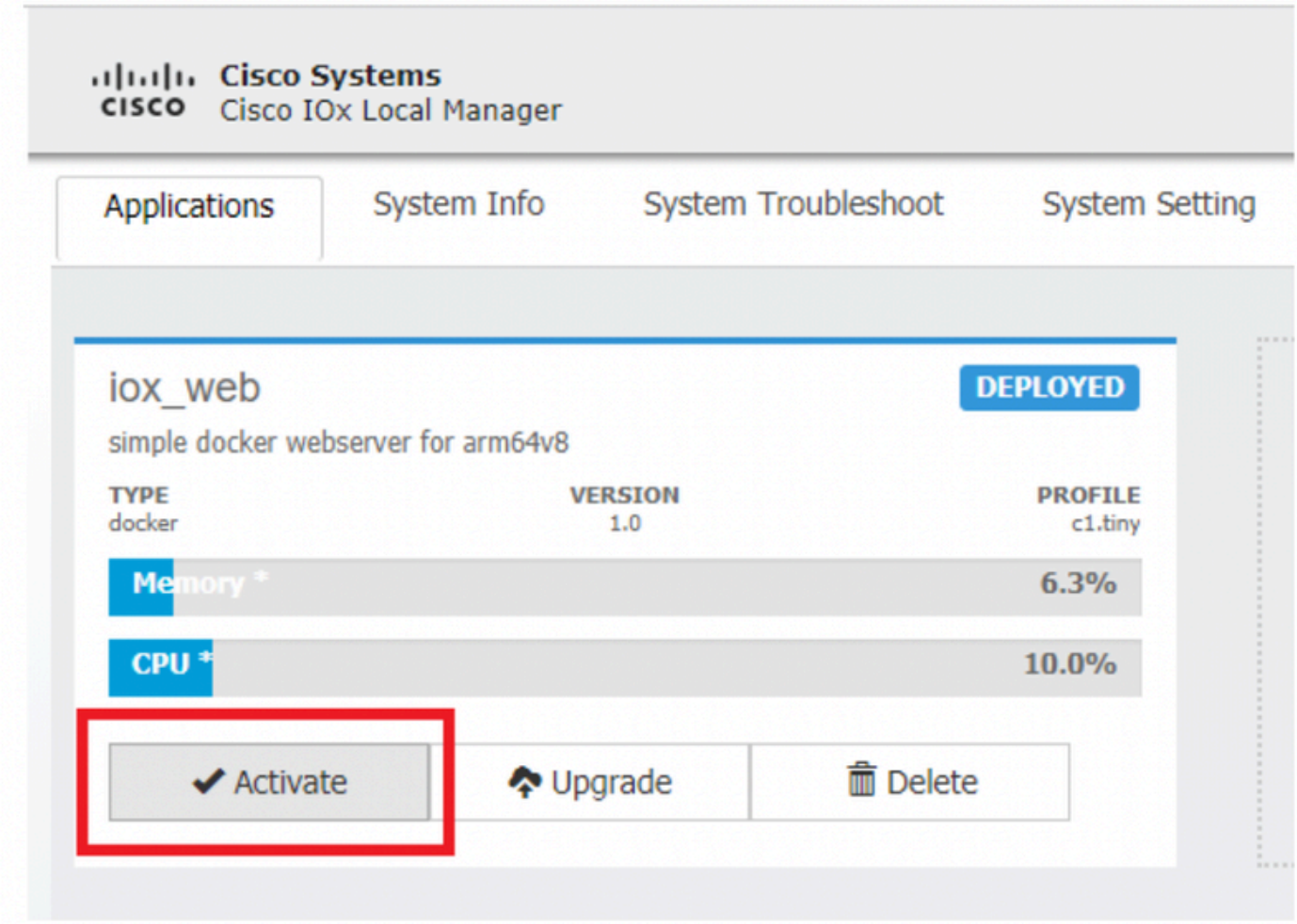
Schritt 3: Verwenden Sie bei der IOx Local Manager-Anmeldung dasselbe Konto, um fortzufahren,

wie im Bild gezeigt:

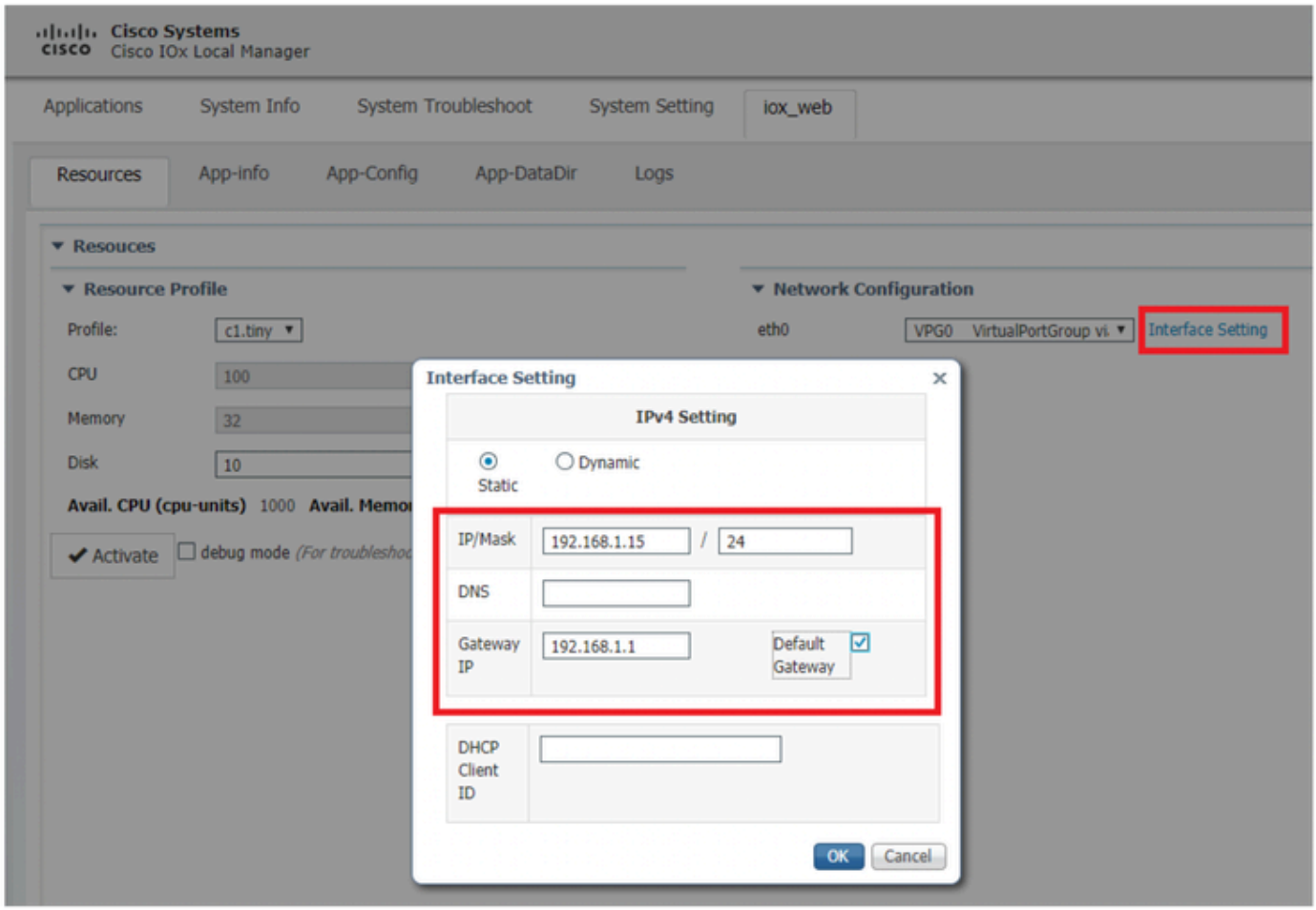
Schritt 4: Klicken Sie auf Add New (Neu hinzufügen), wählen Sie einen Namen für die IOx-Anwendung aus, und wählen Sie die Datei package.tar aus, die in Schritt 3 des Abschnitts Verfahren zum Einrichten einer Buildumgebung mit Vagrant erstellt wurde, wie im Bild gezeigt:



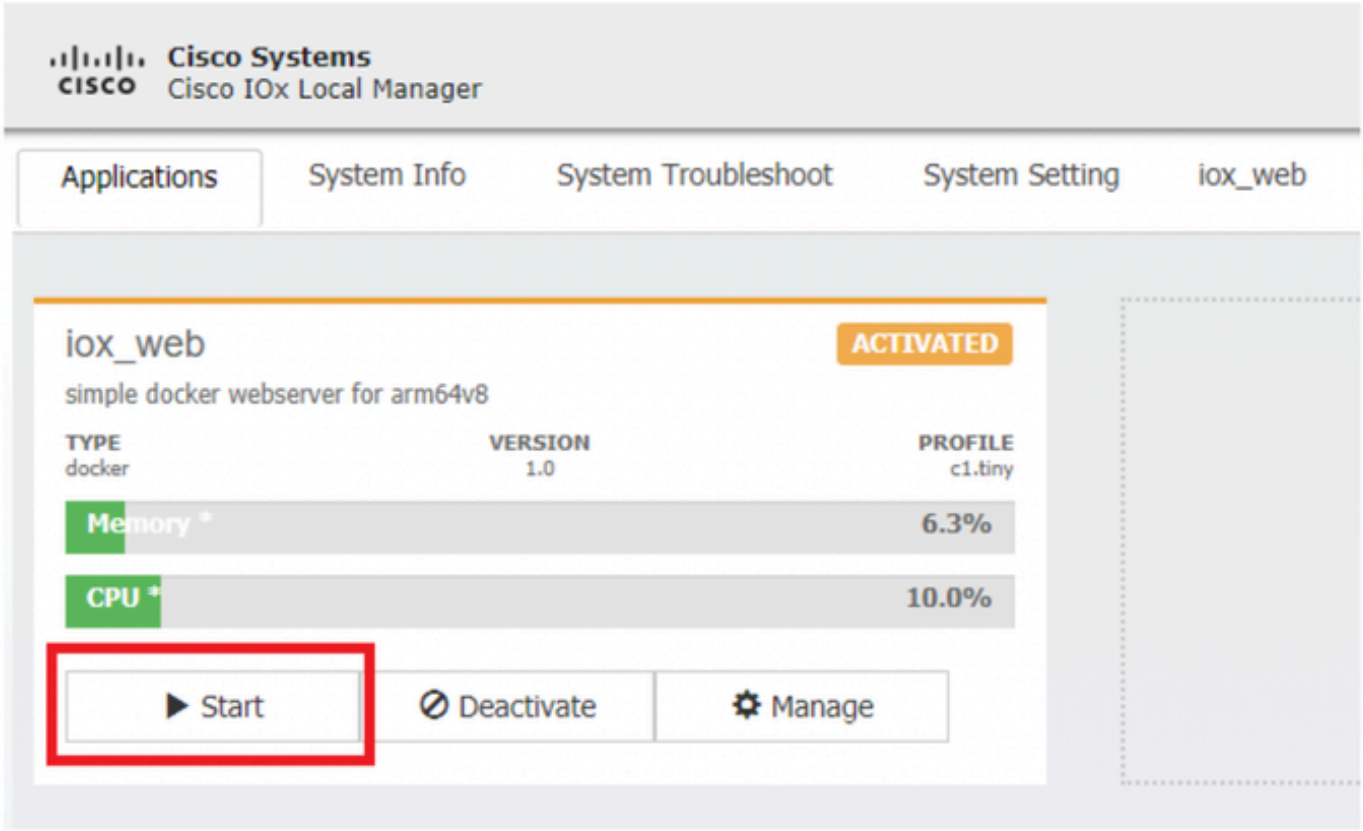
Schritt 5: Sobald das Paket hochgeladen wurde, aktivieren Sie es wie im Bild gezeigt:



Schritt 6: Öffnen Sie auf der Registerkarte Resources die Schnittstelleneinstellung, um die feste IP-Adresse anzugeben, die Sie der App wie im Bild gezeigt zuweisen möchten:



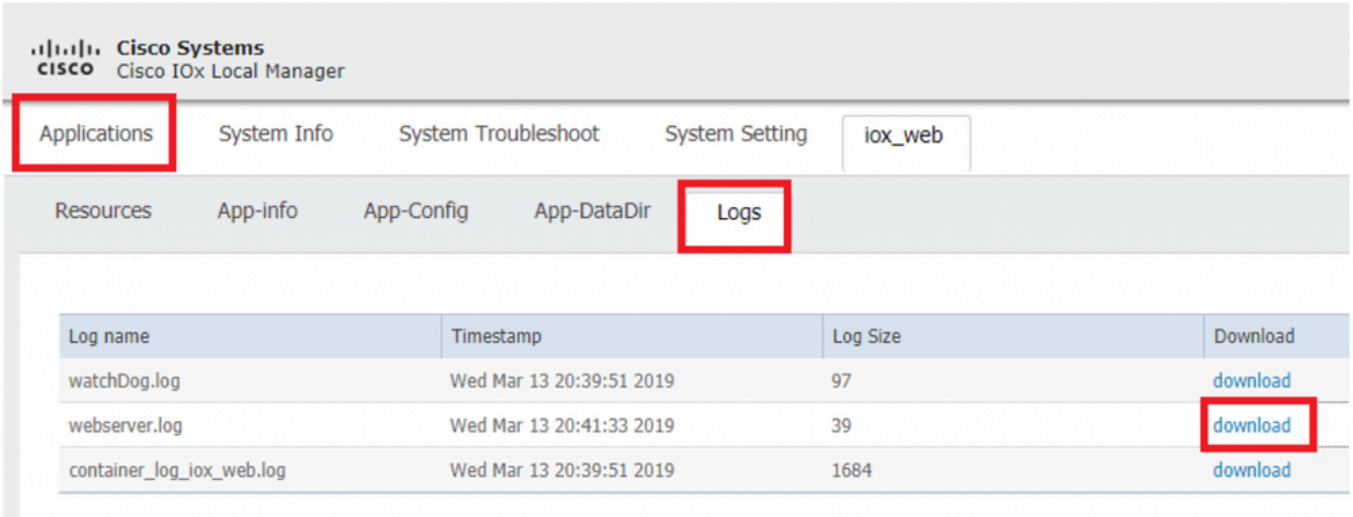
Schritt 7. Klicken Sie auf OK und dann auf Aktivieren. Wenn die Aktion abgeschlossen ist, navigieren Sie zurück zur Hauptseite für den lokalen Manager (Schaltfläche "Anwendungen" im oberen Menü), und starten Sie die Anwendung wie in der Abbildung dargestellt:



Nachdem Sie diese Schritte durchgeführt haben, kann die Anwendung ausgeführt werden.

Fehlerbehebung

Um Probleme mit Ihrer Konfiguration zu beheben, überprüfen Sie die Protokolldatei, die Sie im Python-Skript mit einem lokalen Manager erstellen. Navigieren Sie zu Anwendungen, klicken Sie in der iox_web-Anwendung auf Verwalten, und wählen Sie dann die Registerkarte Protokolle wie im Bild dargestellt aus:



Informationen zu dieser Übersetzung

Cisco hat dieses Dokument maschinell übersetzen und von einem menschlichen Übersetzer editieren und korrigieren lassen, um unseren Benutzern auf der ganzen Welt Support-Inhalte in ihrer eigenen Sprache zu bieten. Bitte beachten Sie, dass selbst die beste maschinelle Übersetzung nicht so genau ist wie eine von einem professionellen Übersetzer angefertigte. Cisco Systems, Inc. übernimmt keine Haftung für die Richtigkeit dieser Übersetzungen und empfiehlt, immer das englische Originaldokument (siehe bereitgestellter Link) heranzuziehen.