Vagrant 및 Virtualbox/VMWare로 IOx 앱 구축

목차

소개

사전 요구 사항

- Windows/MAC Intel/Linux
- MAC ARM 기반 - M1/M2/M3

Vagrant를 사용하여 빌드 환경 설정 절차

- 작업 요약

맞춤형 IOx 애플리케이션 구축 절차

IOx 애플리케이션 구축

문제 해결

소개

이 문서에서는 Vagrant 및 Virtualbox를 사용하여 IOx 애플리케이션을 구축하고 IOx 로컬 관리자 GUI에서 구축하는 방법에 대해 설명합니다.

사전 요구 사항

Windows/MAC Intel/Linux

- Git
- 방랑자
- 가상 상자

MAC ARM 기반 - M1/M2/M3

- Git
- 방랑자
- VMware Fusion(VMware 퓨전)
- vagrant-vmware-desktop 플러그인

다운로드하려면 다음을 수행합니다.

- 방랑자
- 가상 박스

Vagrant를 사용하여 빌드 환경 설정 절차

작업 요약

- Vagrantfile 컨피그레이션은 해당 호스트 머신 아키텍처를 기반으로 VM 환경을 설정합니다.
- 아키텍처에 따라 VMware Fusion 또는 VirtualBox를 사용하도록 VM을 구성합니다
- QEMU(Quick EMUlator), Docker, ioxclient 등 필요한 소프트웨어 및 툴을 사용하여 VM을 프로비저닝합니다.
- 이 구성에서는 amd64 대상 Cisco 플랫폼 장치용 샘플 iperf 애플리케이션을 자동으로 구축합니다.

1단계. 로컬 시스템에서 Github 저장소를 복제합니다.

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

또는 컨피그레이션 엔클로저의 내용을 복사하여 "Vagrantfile"에 붙여넣습니다. 이렇게 하면 로컬 시스템에 "Vagrantfile"이라는 이름의 파일이 생성됩니다.

```
# -*- 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
```

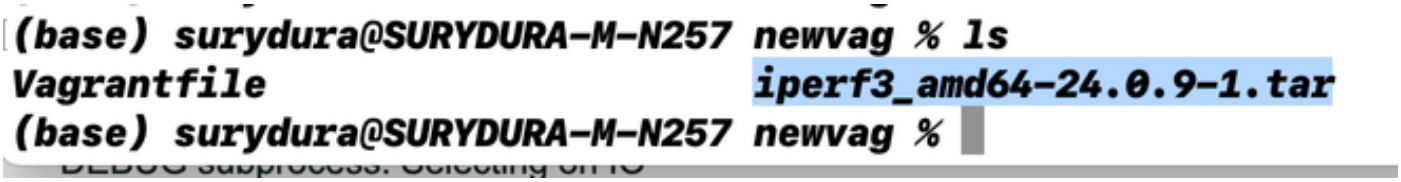
2단계. "export VER="5:24.0.9-1~ubuntu.22.04~jammy" 줄이 주석 처리되어 있지 않고 다른 모든 export 명령문이 주석 처리되어 있는지 확인합니다. 이는 이 Vagrant 환경에 설치하려는 Docker

Engine 버전에 해당합니다.

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

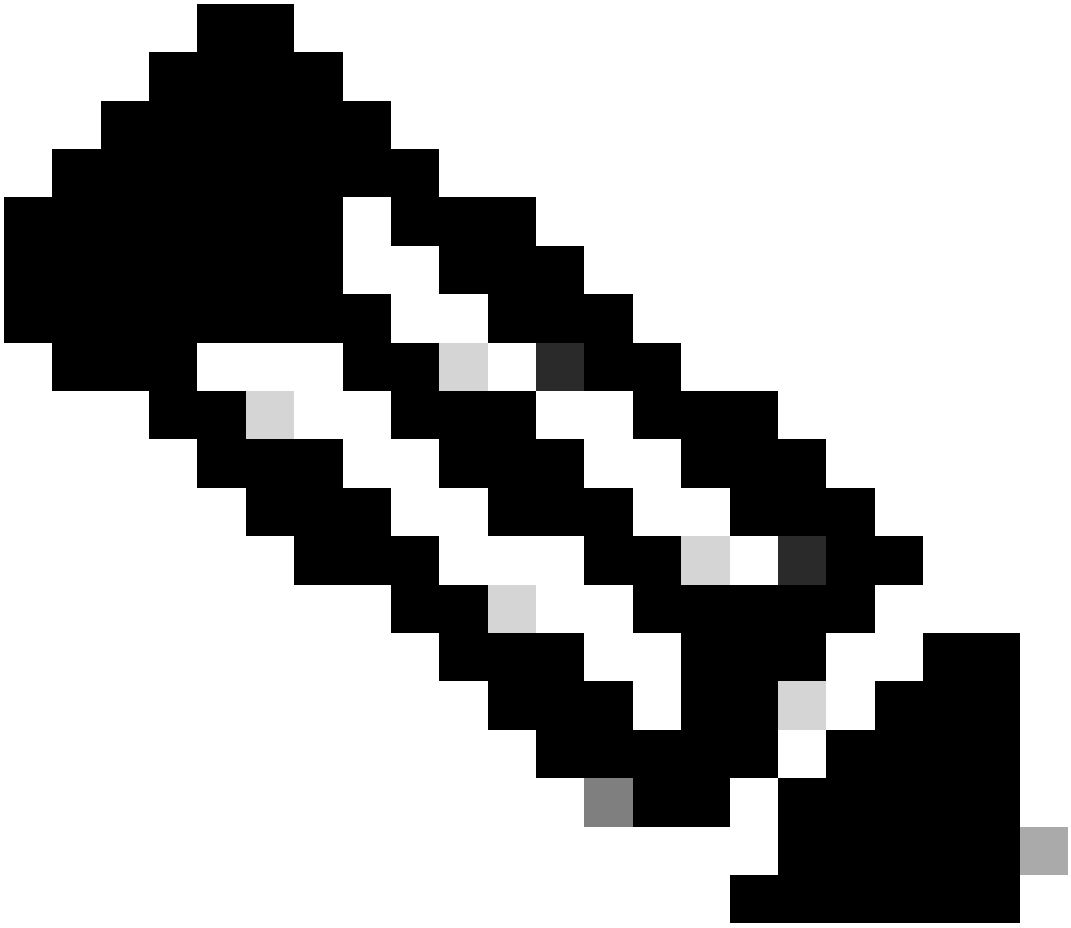
3단계. Vagrantfile이 상주하는 디렉토리에서 vagrant up 명령을 사용하여 Vagrant 환경을 시작하고 amd64 tar 파일에 대한 iperf IOx 애플리케이션의 성공적인 빌드를 관찰합니다.

```
vagrant up
```



맞춤형 IOx 애플리케이션 구축 절차

이 섹션에서는 vagrant 환경을 사용하여 맞춤형 IOx 애플리케이션을 구축하는 방법에 대해 설명합니다.



참고: VM의 "/vagrant" 디렉토리와 호스트 시스템의 "Vagrantfile"이 포함된 디렉토리가 동기화되어 있습니다.

이미지에 표시된 것처럼 new.js 파일은 VM 내에 생성되며 호스트 시스템에서도 액세스할 수 있습니다.

```
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 %
```

1단계. "Vagrantfile"이 있는 동일한 폴더에 샘플 애플리케이션을 복제합니다. 이 예에서는 "iox-multiarch-nginx-nyancat-sample" 애플리케이션이 사용됩니다.

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

2단계. SSH를 사용하여 Vagrant 머신에 로그인합니다.

```
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:~$
```

3단계. 응용 프로그램을 빌드합니다.

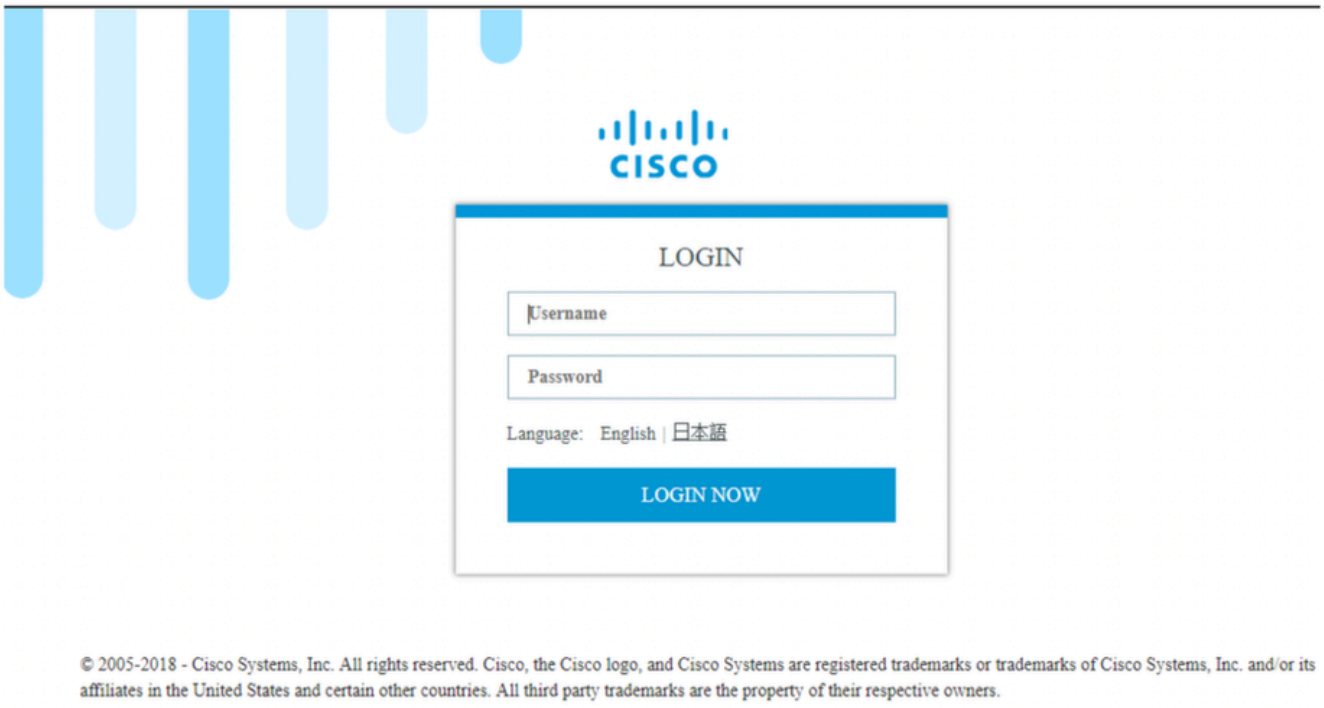
```
cd /vagrant/iox-multiarch-nginx-nyancat-sample/
chmod +x build
sh ./build
```

빌드 프로세스가 완료되면 이제 배포할 준비가 된 두 개의 IOx 응용 프로그램이 준비됩니다 (amd64의 경우 "iox-amd64-nginx-nyancat-sample.tar.gz", 대상 플랫폼의 경우 "iox-arm64-nginx-nyancat-sample.tar.gz").

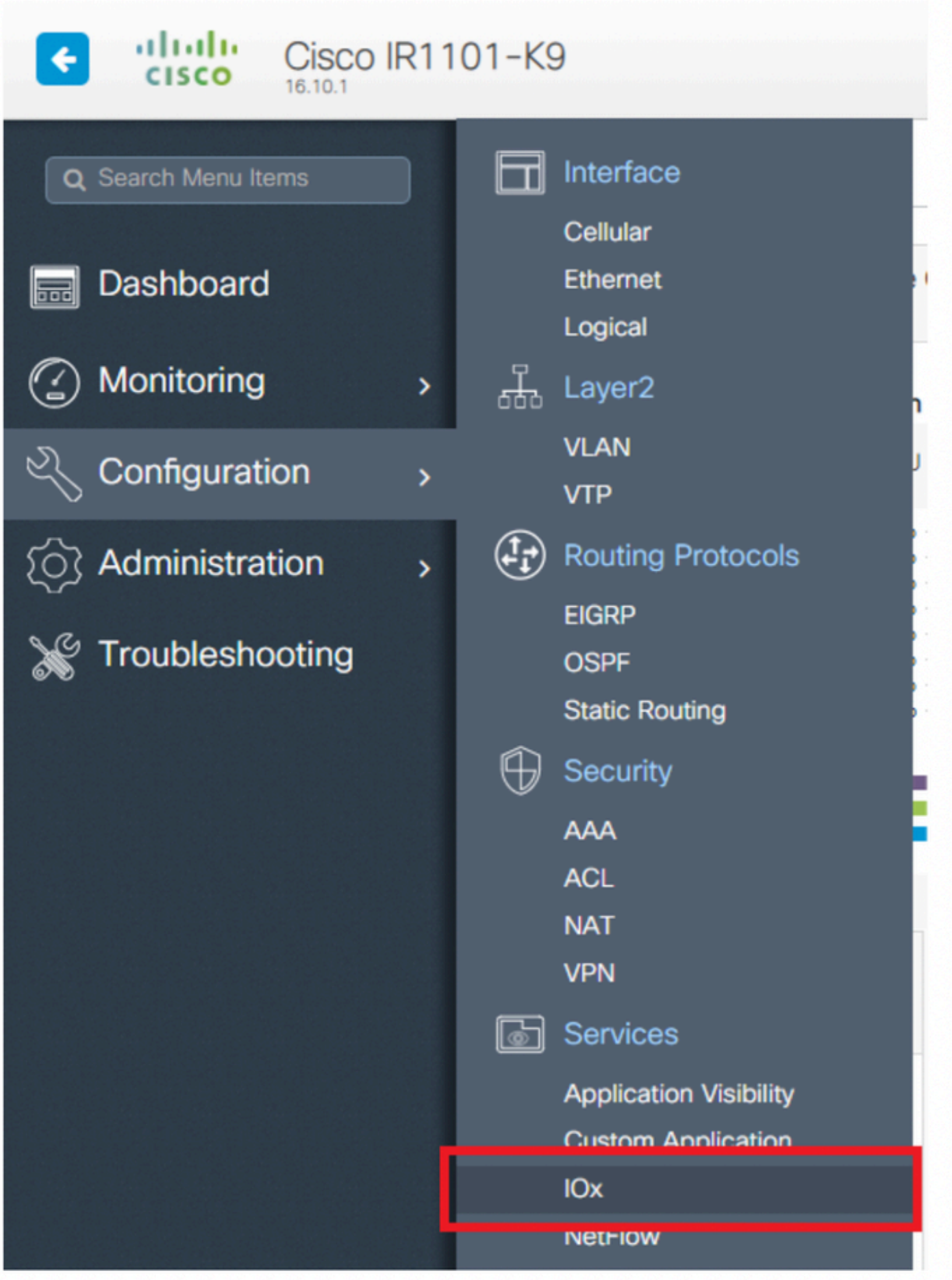
```
Package docker image iox-arm64-nginx-nyancat-sample at /vagrant/iox-multiarch-nginx-nyancat-sample/iox-arm64-nginx-nyancat-sample.tar.gz
vagrant@vagrant:/vagrant/iox-multiarch-nginx-nyancat-sample$ ls
Dockerfile  README.md  images                          iox-arm64-nginx-nyancat-sample.tar.gz  nyan-cat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

IOx 애플리케이션 구축

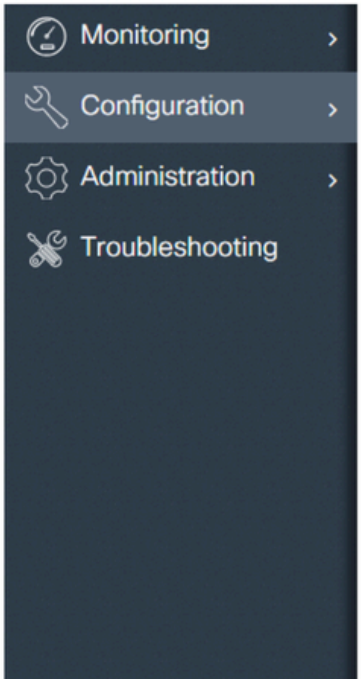
1단계. 웹 인터페이스를 사용하여 IR1101에 액세스합니다.

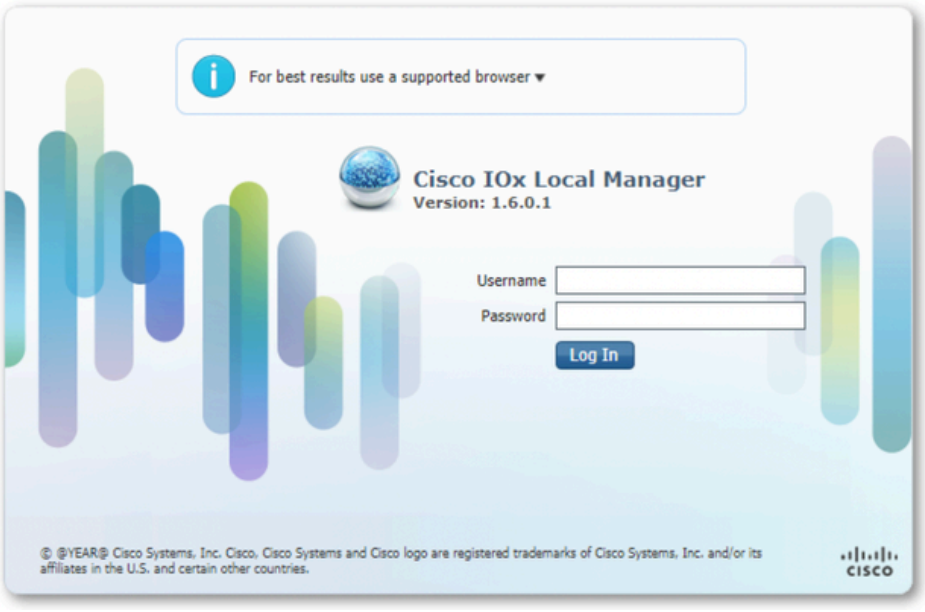


2단계. 권한 15 계정 사용:

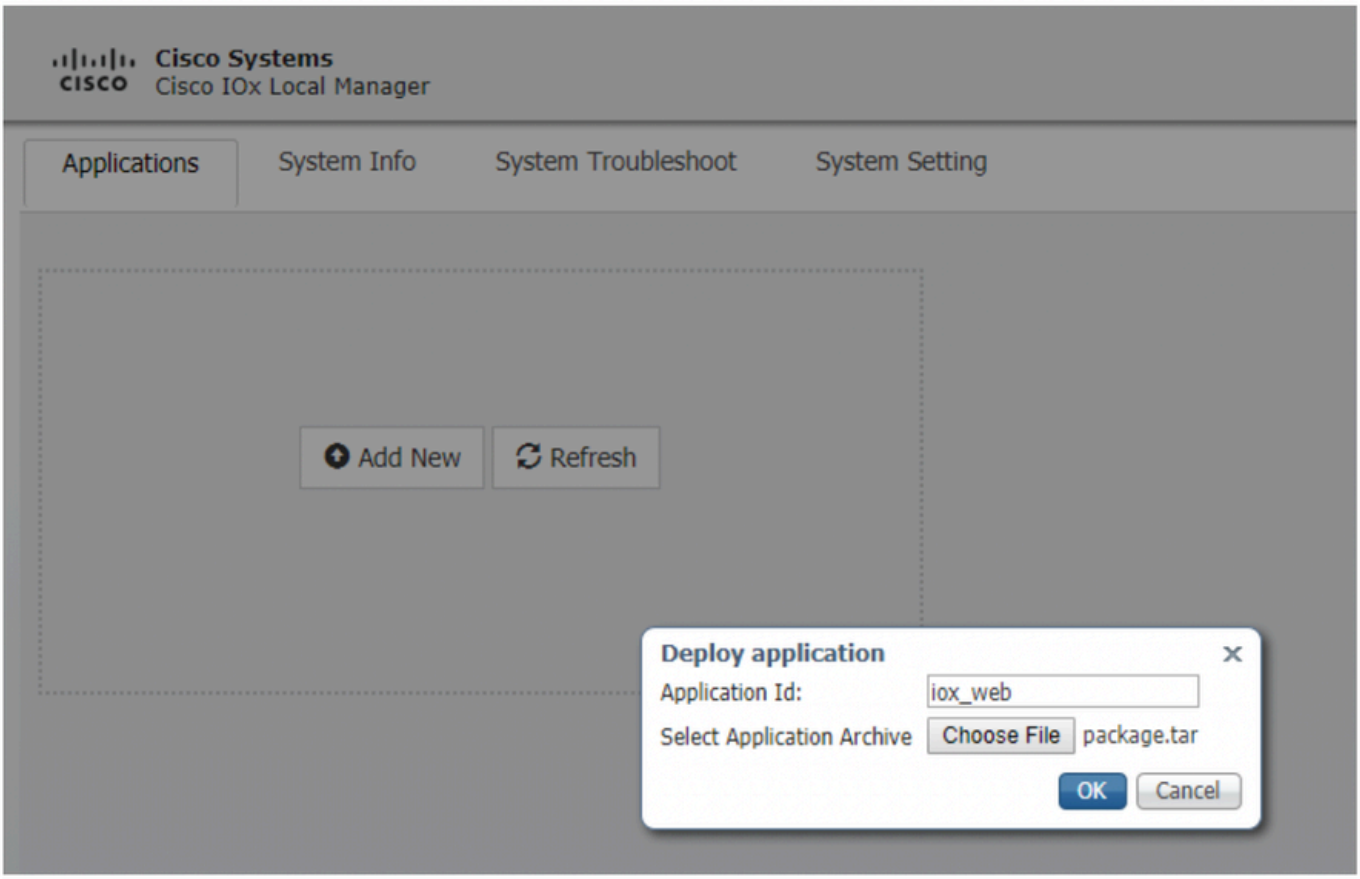


3단계. IOx Local Manager 로그인에서 동일한 계정을 사용하여 그림과 같이 계속 진행합니다.

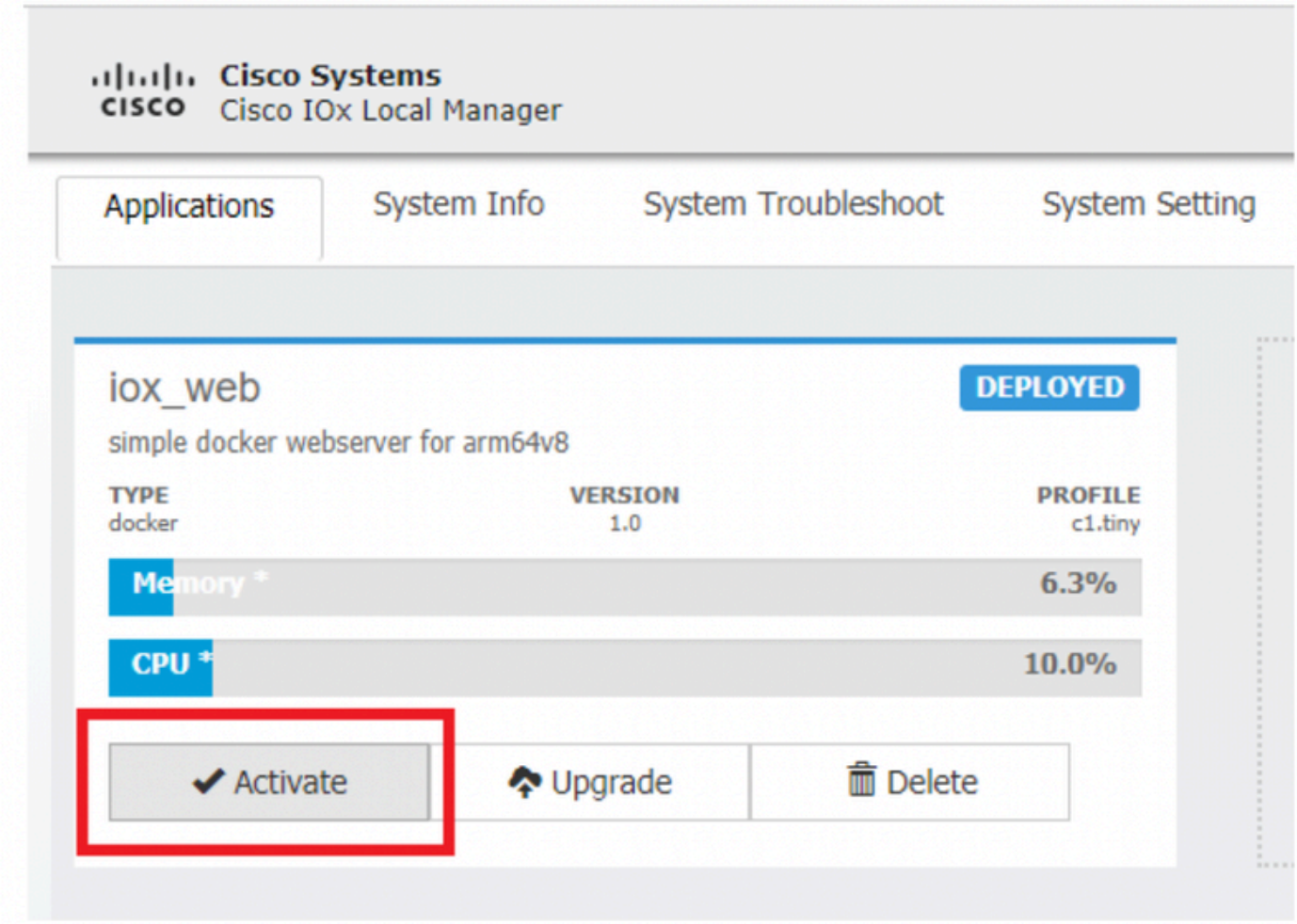




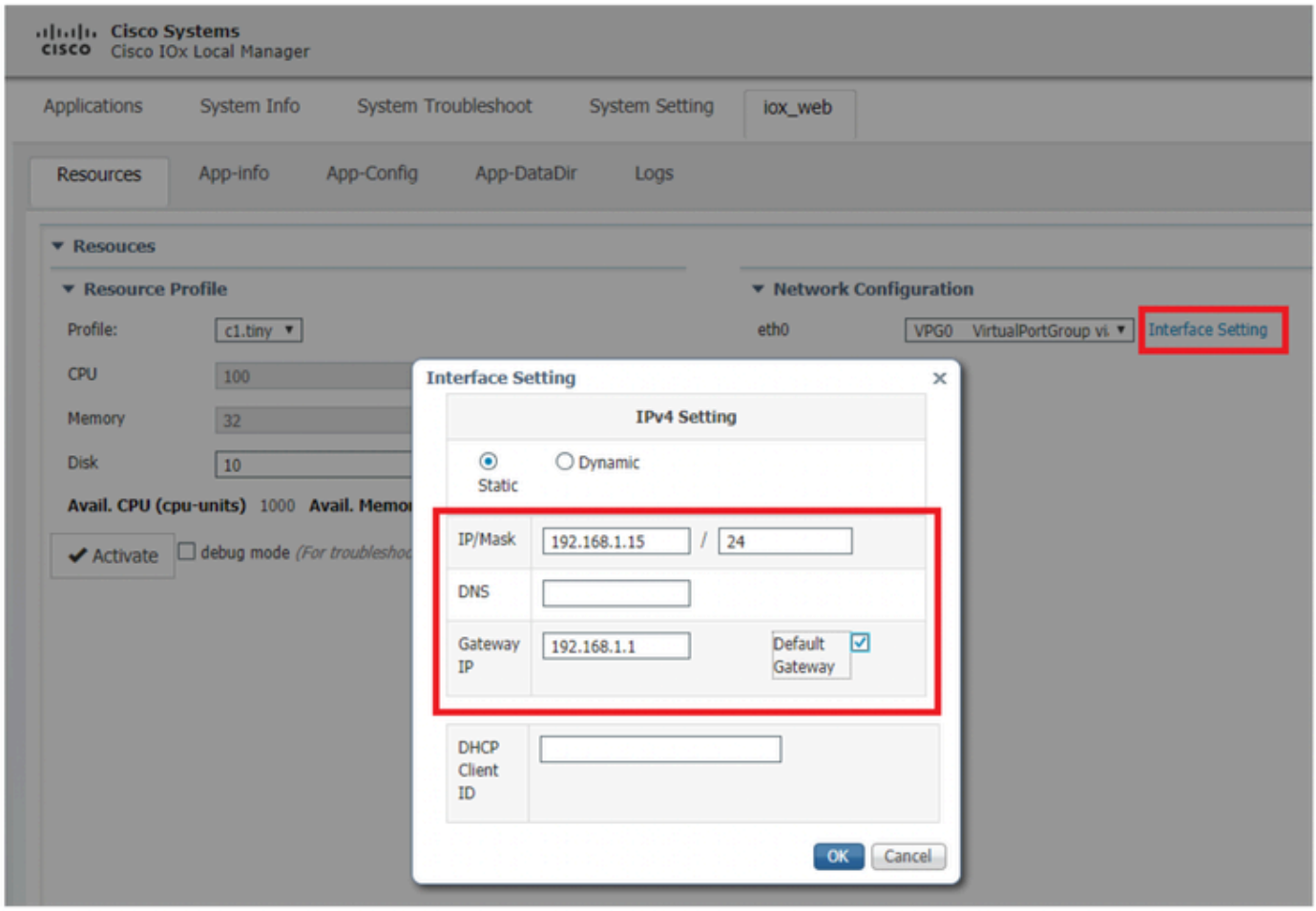
4단계. Add New(새로 추가)를 클릭하고 IOx 애플리케이션의 이름을 선택한 다음 Package.tar(Vagrant를 사용하여 빌드 환경 설정 절차)의 3단계에서 빌드한 패키지를 선택합니다.



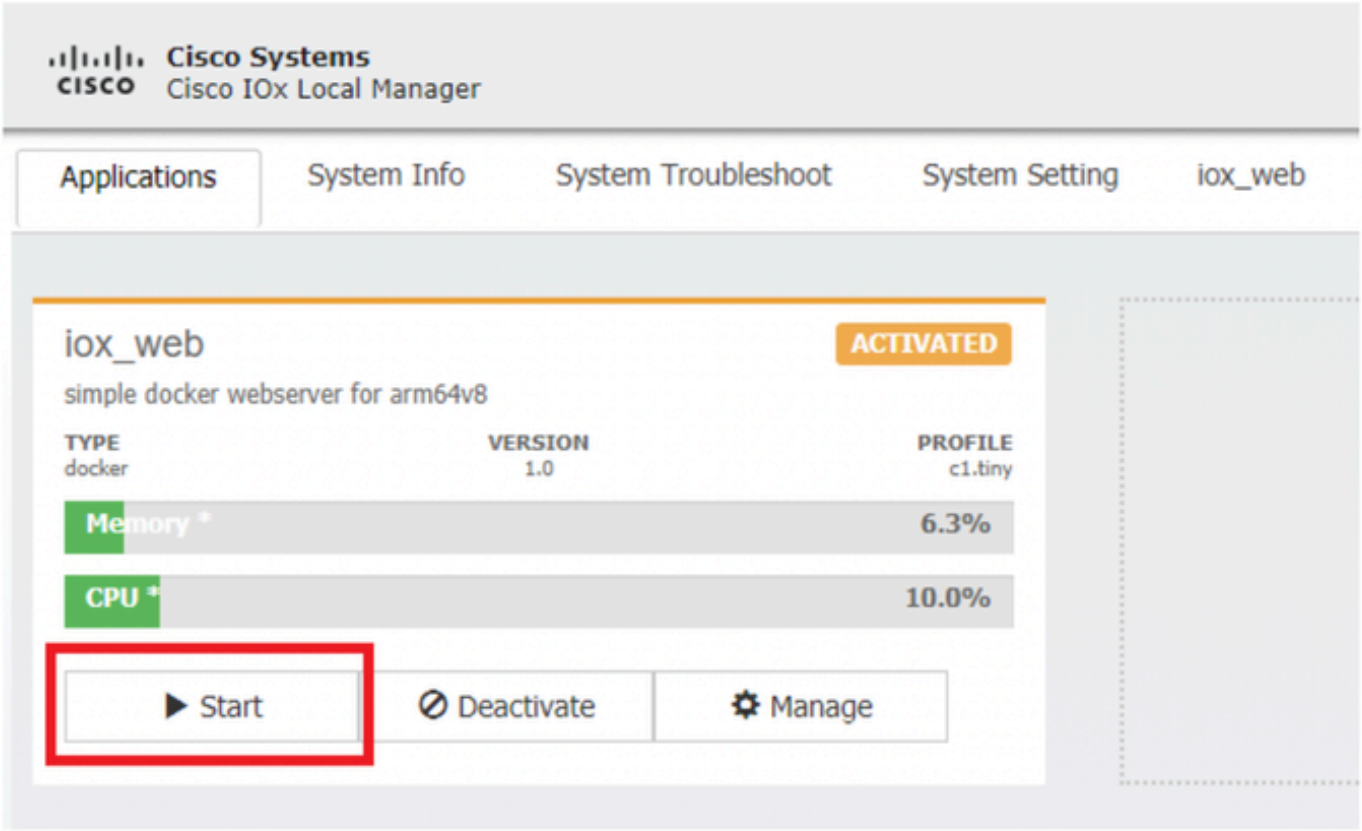
5단계. 패키지가 업로드되면 이미지에 표시된 대로 활성화하십시오.



6단계. 이미지에 표시된 대로 앱에 할당할 고정 IP를 지정하려면 Resources(리소스) 탭에서 인터페이스 설정을 엽니다.



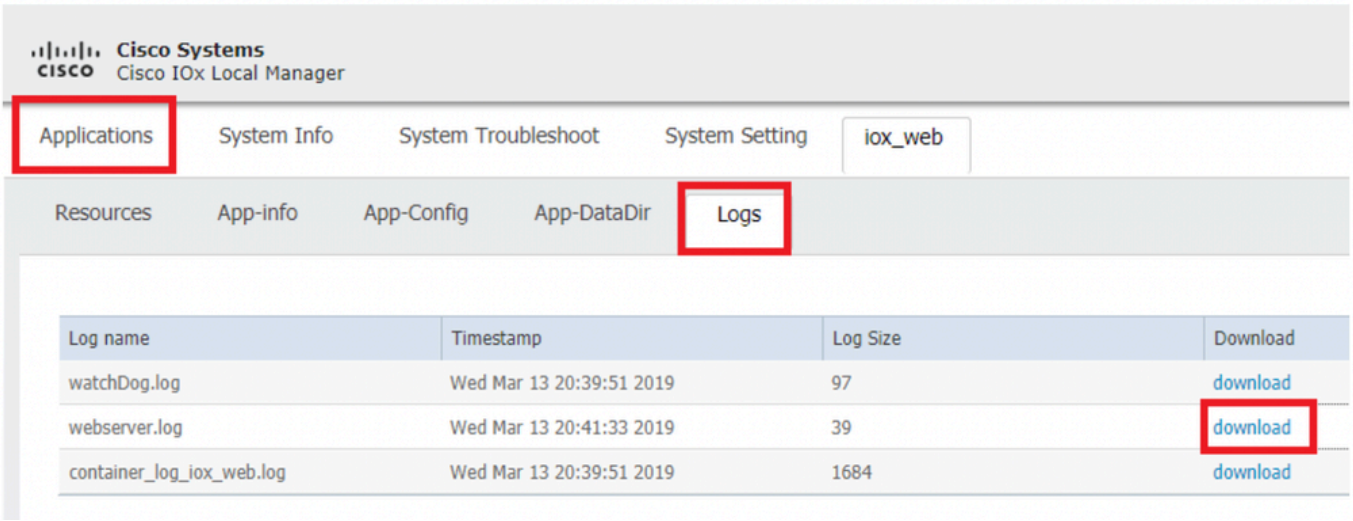
7단계. OK(확인), Activate(활성화)를 차례로 클릭합니다. 작업이 완료되면 기본 로컬 관리자 페이지(상단 메뉴의 응용 프로그램 단추)로 다시 이동한 다음 이미지에 표시된 대로 응용 프로그램을 시작합니다.



이 단계를 거친 후에는 응용 프로그램을 실행할 준비가 됩니다.

문제 해결

컨피그레이션의 문제를 해결하려면 로컬 관리자를 사용하여 Python 스크립트에서 생성하는 로그 파일을 확인하십시오. Applications(애플리케이션)로 이동하고 iox_web 애플리케이션에서 Manage(관리)를 클릭한 다음 이미지에 표시된 대로 Logs(로그) 탭을 선택합니다.



이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.