Configure Windows VM to CGM-SRV Module on CGR1xxx

Contents

[Introduction](#)
[Prerequisites](#)
[Requirements](#)
[Components Used](#)
[Background Information](#)
[Configure](#)
[Create a Windows VM Image](#)
[Install KVM on Your Linux Machine](#)
[Verify KVM Installation](#)
[Create a Windows VM](#)
[Deploy Windows VM Image to CGM-SRV](#)
[Verify](#)
[Troubleshoot](#)

Introduction

This document describes the necessary steps in order to create and run a Windows Virtual Machine (VM) on the Connected Grid Module (CGM) - System Server (SRV) module.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- Linux
- Kernel Based Virtual Machine (KVM)
- Understand Virtualization concepts

Components Used

The information in this document is based on these software and hardware versions:

- Connected Grid Routers (CGR) 1120
- CGM-SRV-XX module
- Configuration steps for CGM-SRV are executed prior to this guide:
- Windows 7 installation ISO
- Virtual Network Computing (VNC) viewer

The information in this document was created from the devices in a specific lab environment. All of

the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Background Information

When you want to run IOx applications or VMs on the CGR1000 platform, you can use the CGM-SRV compute module. The CGM-SRV module is actually a small server that contains a multi-core x86 CPU, memory and storage. Both the CGR1120 and CGR1240 can have one of these modules to add IOx capabilities.

There are, at the time of writing, two types available:

Stock Keep Unit (SKU)	Solid State Drives (SSD)	RAM	CPU
CGM-SRV-64	64GB (50GB usable)	4GB	4 core 800Mhz
CGM-SRV-128	128GB (100GB usable)	4GB	4 core 800Mhz

Each module also has two USB ports for storage and its own external gigabit Ethernet interface.

As with any other IOx-capable device, the module can host different types of IOx applications but due to the larger capacity of the CGM-SRV module, it can also run a fully configured Windows or standard Linux distro (for example Ubuntu or CentOS).

Configure

Create a Windows VM Image

In order to deploy a Windows VM on the CGM-SRV module, you first need to create an image in the QEMU QCOW format which contains the Windows installation. One way to create such an image is with KVM and virsh on a Linux machine.

The steps mentioned further do not involve the CGR1xxx or CGM-SRV at all, they are just required steps to create a basic Windows 7 VM QCOW image which you can deploy in the next step to the CGM-SRV.

For this guide, you can start with a freshly install CentOS7 minimal installation. The steps for other Linux distributions must be similar but can slightly differ.

Install KVM on Your Linux Machine

Step 1. The first thing to do is to check if the host-machine supports VM-extensions. On the x86 platform, those are either AMD-V or Intel's VT-X. Most, if not all, modern x86 CPUs support these extensions. Even when you run a VM, most hypervisors provide the option to pass/emulate these extensions.

In order to check if the installed CPU's support those extensions, you need to check if the vmx (for VT-X) or svm (for AMD-V) flag exists in the cpuinfo-output.

```
[root@cen7 ~]# egrep -c '(vmx|svm)' /proc/cpuinfo
2
```

If the output of this command is 0, this means that no CPU found supports the VM-extensions. In that case, you can check if these extensions are enabled in your BIOS or hypervisor when you use a VM to run this machine.

Step 2. The next step is to create a bridge to provide a network for the VM which you can run on KMV.

Firstly, you need to enable IP forwards in the kernel:

```
[root@cen7 ~]# echo "net.ipv4.ip_forward = 1"|sudo tee /etc/sysctl.d/99-ipforward.conf
net.ipv4.ip_forward = 1
[root@cen7 ~]# sysctl -p /etc/sysctl.d/99-ipforward.conf
net.ipv4.ip_forward = 1
```

In order to create the bridge, the IP configuration needs to move from the real interface to the bridge itself, as this is the interface that owns the IP address.

After you complete a standard installation, the network configuration is in **/etc/sysconfig/network-scripts**:

```
[root@cen7 ~]# ls -1 /etc/sysconfig/network-scripts/ifcfg-*
/etc/sysconfig/network-scripts/ifcfg-eno16777736
/etc/sysconfig/network-scripts/ifcfg-lo
```

Step 3. As you can see, there is currently one interface (besides the loopback interface), called eno167777736. You need to move the IP-related configuration to a bridge interface which you can call virbr0:

```
[root@cen7 ~]# vi /etc/sysconfig/network-scripts/ifcfg-virbr0
[root@cen7 ~]# cat /etc/sysconfig/network-scripts/ifcfg-virbr0
DEVICE=virbr0
TYPE=BRIDGE
ONBOOT=yes
BOOTPROTO=static
IPADDR=172.16.245.162
NETMASK=255.255.255.0
GATEWAY=172.16.245.2
DNS1=8.8.8.8
```

Step 4. After that, you need to clean up the IP configuration from the real interface and connect it to the virbr0 bridge:

```
[root@cen7 ~]# vi /etc/sysconfig/network-scripts/ifcfg-eno16777736
[root@cen7 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eno16777736
UUID=46f0f247-e164-40cc-866b-9133458d9df8
DEVICE=eno16777736
ONBOOT=yes
BRIDGE=virbr0
HWADDR=00:0c:29:ce:96:38
```

Step 5. Once the network configuration is complete, you can go ahead and install KVM:

```
[root@cen7 ~]# sudo yum install kvm virt-manager libvirt virt-install qemu-kvm xauth dejavu-lgc-sans-fonts -y
...
Complete!
```

Step 6. After the installation is complete, the best is to reboot this machine to apply the newly installed modules and network configuration:

```
[root@cen7 ~]# init 6
```

Verify KVM Installation

Step 7. After the reboot has completed, you should be able to access the machine on the (same) IP configured on the bridge interface. You must check if the KVM kernel module is loaded:

```
root@cen7 ~]# lsmod|grep kvm
kvm_intel             200704  0
kvm                   589824  1 kvm_intel
irqbypass              16384  1 kvm
```

Step 8. If this looks fine, you can try to connect with virsh:

```
[root@cen7 ~]# sudo virsh -c qemu:///system list
 Id    Name                           State
----------------------------------------------------
```

Step 9. One last step is to open port 5900 on the firewall on this machine for VNC access to the Windows installation:

```
[root@cen7 ~]# firewall-cmd --zone=public --add-port=5900/tcp --permanent
success
[root@cen7 ~]# firewall-cmd --reload
success
```

Create a Windows VM

Now that you have a system which works with KVM installation, you can fire up a new VM on KVM and run through the Windows installation dialogs.

Step 1. Copy the Windows 7 installation ISO to your VM (or make it accessible over the network):

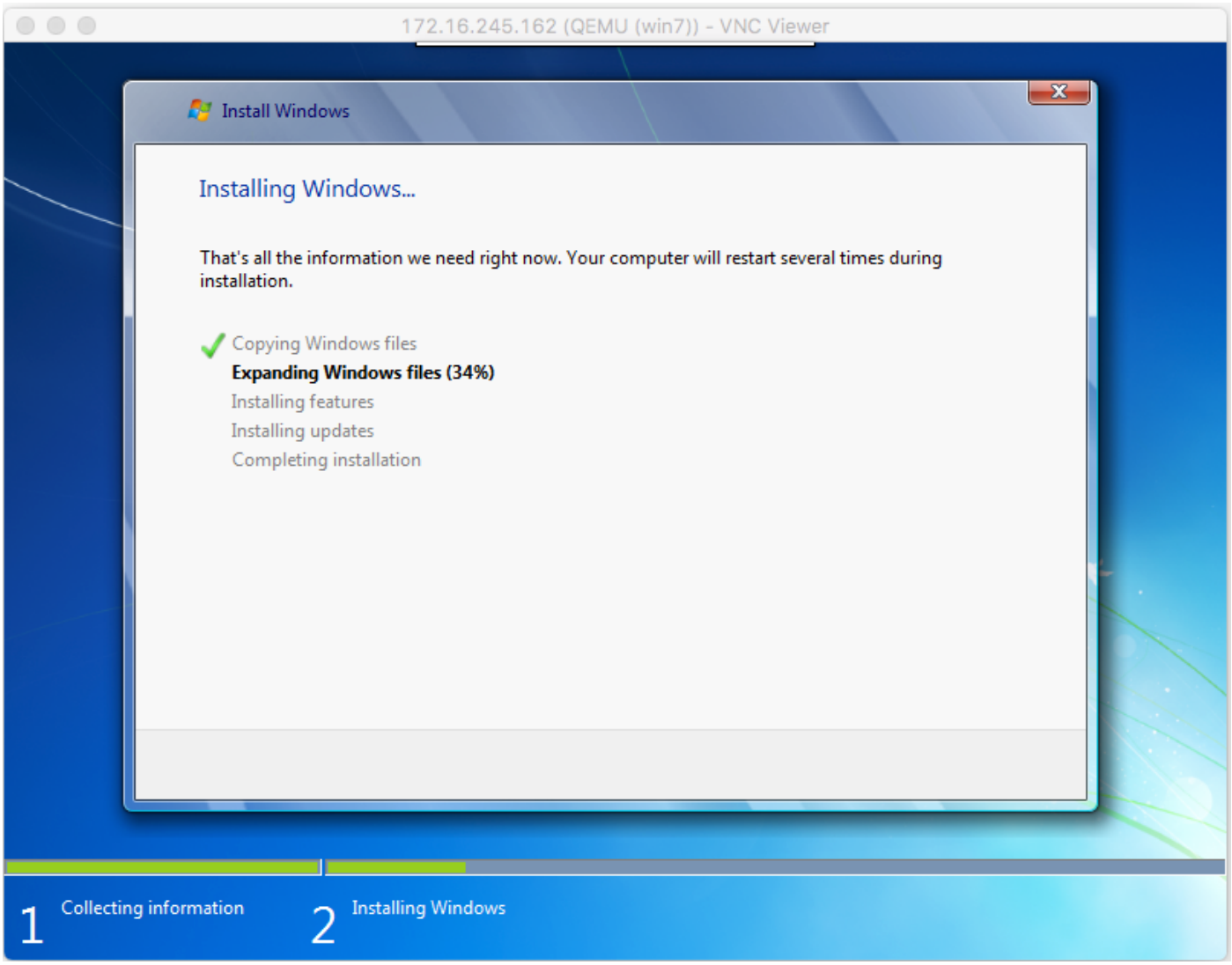
```
[root@cen7 ~]# scp jedepuyd@172.16.X.X:/home/jedepuyd/win7install.iso /var
jedepuyd@172.16.X.X's password:
win7install.iso                                               100% 4546MB  62.1MB/s
01:13
```

Step 2. Create a new KVM VM and let it boot from the Windows 7 ISO:

```
root@cen7 ~]# virt-install --connect qemu:///system -n win7 -r 1024 --vcpus=2 --disk
path=/var/lib/libvirt/images/win7.img,size=9 --graphics vnc,listen=0.0.0.0 --noautoconsole --os-
type windows --os-variant win7 --accelerate --network=bridge:virbr0 --hvm --cdrom
/var/win7install.iso

Starting install...
Allocating win7.img                                                 | 9.0 GB
00:00:00
Creating domain...                                                    |    0 B
00:00:00
Domain installation still in progress. You can reconnect to
the console to complete the installation process.
```

Step 3. Once the VM has started, you can connect with the use of VNC viewer to the IP of the host machine on port 5900 and finish the standard Windows installation as shown in the image:



If Windows reboot at the time of the installation, it could be necessary to restart the VM with virsh if this isn't done automatically:

```
[root@cen7 ~]# virsh start win7
Domain win7 started
```

Step 4. Once the installation has completed, shut down the VM. You now have a QCOW-image of this installation in the path provided when you create the VM: **/var/lib/libvirt/images/win7.img**. This type of image can be deployed on the CGM-SRV to run Windows.

Deploy Windows VM Image to CGM-SRV

Now that you have the correct type of image to run on the CGM-SRV, you can start to deploy it.

Step 1. Setup a profile for ioxlcient that corresponds with your configuration:

```
[root@cen7 ~]# ./ioxclient profiles create
Enter a name for this profile : CGR1120_20
Your IOx platform's IP address[127.0.0.1] : 10.x.x.x.x
Your IOx platform's port number[8443] :
```

```
Authorized user name[root] : admin
Password for admin :
Local repository path on IOx platform[/software/downloads]:
URL Scheme (http/https) [https]:
API Prefix[/iox/api/v2/hosting/]:
Your IOx platform's SSH Port[2222]:
Your RSA key, for signing packages, in PEM format[]:
Your x.509 certificate in PEM format[]:
Activating Profile  CGR1120_20
Saving current configuration
```

In this example, 10.X.X.X corresponds with the outgoing interface on the CGR1000 on which you configured Network Address Translation (NAT) to forward to port 8443 on the CGM-SRV.

Step 2. Now that ioxclient is configured, let's rename your earlier created image to **vm.img** in order to simplify a bit and copy it with the use of Secure Copy (SCP) with ioxclient to CGM-SRV.

Optionally, convert the disk image to the QCOW2 format as that is what the CGM-SRV is expecting. Newer versions of virt-manager seem to create the disk images by default in the QCOW3 format.

You can easily convert the image with the use of this command:

```
[root@cen7 ~]# qemu-img convert -f qcow2 -O qcow2 /var/lib/libvirt/images/win7.img
/var/lib/libvirt/images/win7.img
```

Once you are sure that the image is in the right format, proceed with the rename and copy:

```
[root@cen7 ~]# mv /var/lib/libvirt/images/win7.img /root/vm.img
[root@cen7 ~]# ./ioxclient platform scp /root/vm.img
Currently active profile :  CGR1120_20
Command Name:  plt-scp
Saving current configuration
Downloaded scp keys to pscp.pem
Running command : [scp -P 2222 -r -i pscp.pem /root/vm.img scpuser@10.50.215.246:/]
```

This transfer could take a while, the transfer rates from around 3-4MB/s to the CGM-SRV via Cisco IOS®. The file gets copied to **/mnt/data/vm/vm.img** on the CGM-SRV module.

Step 3. While the transfer is in progress (or complete), you can create the **package.yaml** file. This file describes to IOx what exactly you would like to deploy and how to package it.

```
[root@cen7 ~]# vi package.yaml
[root@cen7 ~]# cat package.yaml
descriptor-schema-version: 2.2

info:
  author-link: http://www.cisco.com/ author-name: Jens Depuydt description: Windows 7 VM for
CSR-SRV name: win7 version: 1.0 app: type: vm cpuarch: x86_64 resources: profile: custom cpu:
600 disk: 10 memory: 3072 network: - interface-name: eth0 - interface-name: eth1 graphics: vnc:
true startup: ostype: windows qemu-guest-agent: false disks: - target-dev: hda file:
file://vm.img
```

As you can see in this **package.yaml**, you refer to **file://vm.img** which corresponds with the real location of mnt/data/vm/vm.img on the CGM-SRV module.

Step 4. The next step is to package with the use of ioxclient:

```
[root@cen7 ~]# ./ioxclient pkg .
Currently active profile :  default
Command Name:  package
No rsa key and/or certificate files to sign the package
Checking if package descriptor file is present..
Validating descriptor file /root/package.yaml with package schema definitions
Parsing descriptor file..
Found schema version  2.2
Loading schema file for version  2.2
Validating package descriptor file..
File /root/package.yaml is valid under schema version 2.2
Created Staging directory at :  /var/folders/sp/f9qn2fsn0d5fkj7szps6qvvr0000gn/T/638513626
Copying contents to staging directory
Checking for application runtime type
Couldn't detect application runtime type
Creating an inner envelope for application artifacts
Excluding  .DS_Store
Generated  /var/folders/sp/f9qn2fsn0d5fkj7szps6qvvr0000gn/T/638513626/artifacts.tar.gz
Calculating SHA1 checksum for package contents..
Package MetaData file was not found at
/private/var/folders/sp/f9qn2fsn0d5fkj7szps6qvvr0000gn/T/638513626/.package.metadata
Wrote package metadata file :
/private/var/folders/sp/f9qn2fsn0d5fkj7szps6qvvr0000gn/T/638513626/.package.metadata
Root Directory :  /private/var/folders/sp/f9qn2fsn0d5fkj7szps6qvvr0000gn/T/638513626
Output file:  /var/folders/sp/f9qn2fsn0d5fkj7szps6qvvr0000gn/T/559089521
Path:  .package.metadata
SHA1 : 262f763740c182f95358be84514a76ac11e37012
Path:  artifacts.tar.gz
SHA1 : 3d89ccd35fe5318dd83a249a26cb8140d98d15bb
Path:  package.yaml
SHA1 : aa42f949b707df07a83a17344e488c44eb585561
Generated package manifest at  package.mf
Generating IOx Package..
Package generated at /root/package.tar
```

Step 5. After you create the package, you can install it on our CGM-SRV. The IOx application/VM is called win7 in this example:

```
[root@cen7 ~]# ./ioxclient app install win7 package.tar
Currently active profile :  default
Command Name:  application-install
Saving current configuration

Installation Successful. App is available at :
https://10.X.X.X:8443/iox/api/v2/hosting/apps/win7 Successfully deployed
```

Step 6. Before you can active the win7 IOx VM, you need to create a payload JSON-file that sets the VNC password for this VM:

```
[root@cen7 ~]# vi vnc.json
[root@cen7 ~]# cat vnc.json
{
    "resources": {
    "graphics": {"vnc-password": "password"}
    }
}
```

Step 7. With the use of the **vnc.json** payload, you can activate the win7 IOx VM:

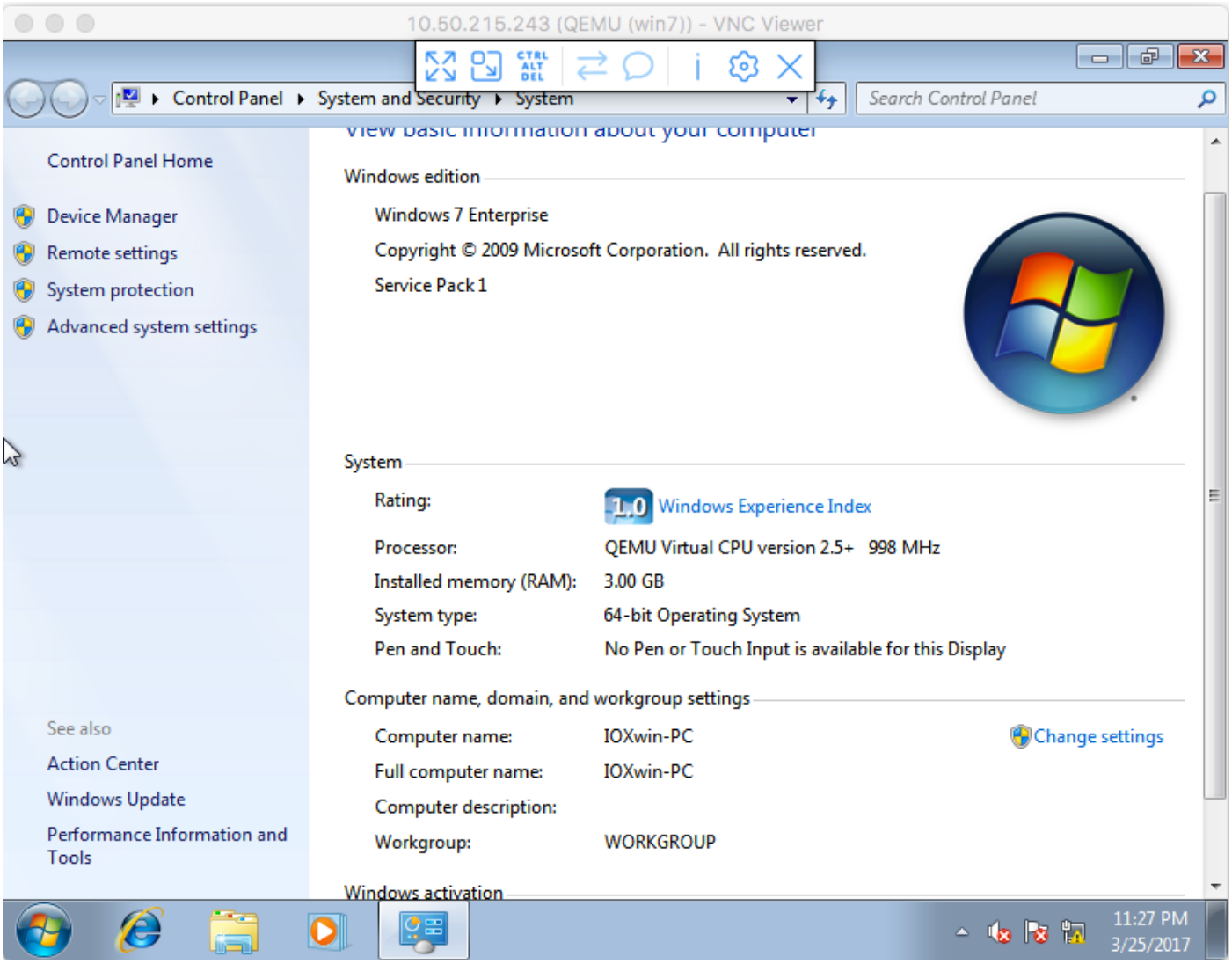
```
[root@cen7 ~]# ./ioxclient app activate win7 --payload vnc.json
Currently active profile :  default
Command Name:  application-activate
Payload file : vnc.json. Will pass it as application/json in request body..
App win7 is Activated
```

Step 8. Last step with ioxclient is to start the VM:

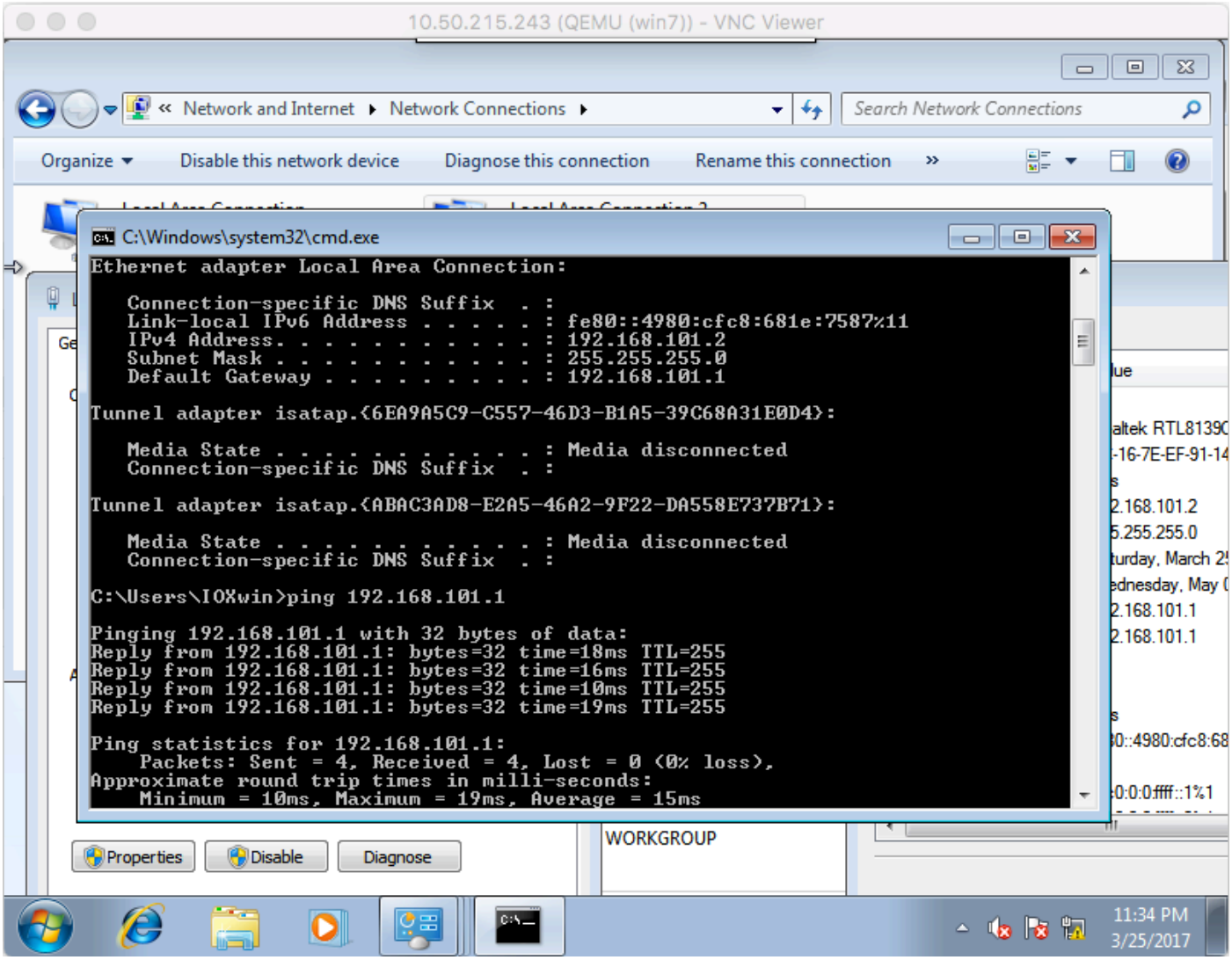
```
[root@cen7 ~]# ./ioxclient app start win7
Currently active profile :  default
Command Name:  application-start
App win7 is Started
```

At this point, the Windows VM runs on the CGM-SRV and you can start to use it.

In order to get access to the Windows machine console, you can use VNC viewer on the outgoing interface on the CGR1000 and port 5900 as shown in the image:



From a network perspective, you chose to give eth0 and eth1 to the win7 IOx VM with the use of the **package.yaml** file as shown in the image:



As you can see, these interfaces got an IP from the DHCP server which runs on Cisco IOS® and can be used without further configuration.

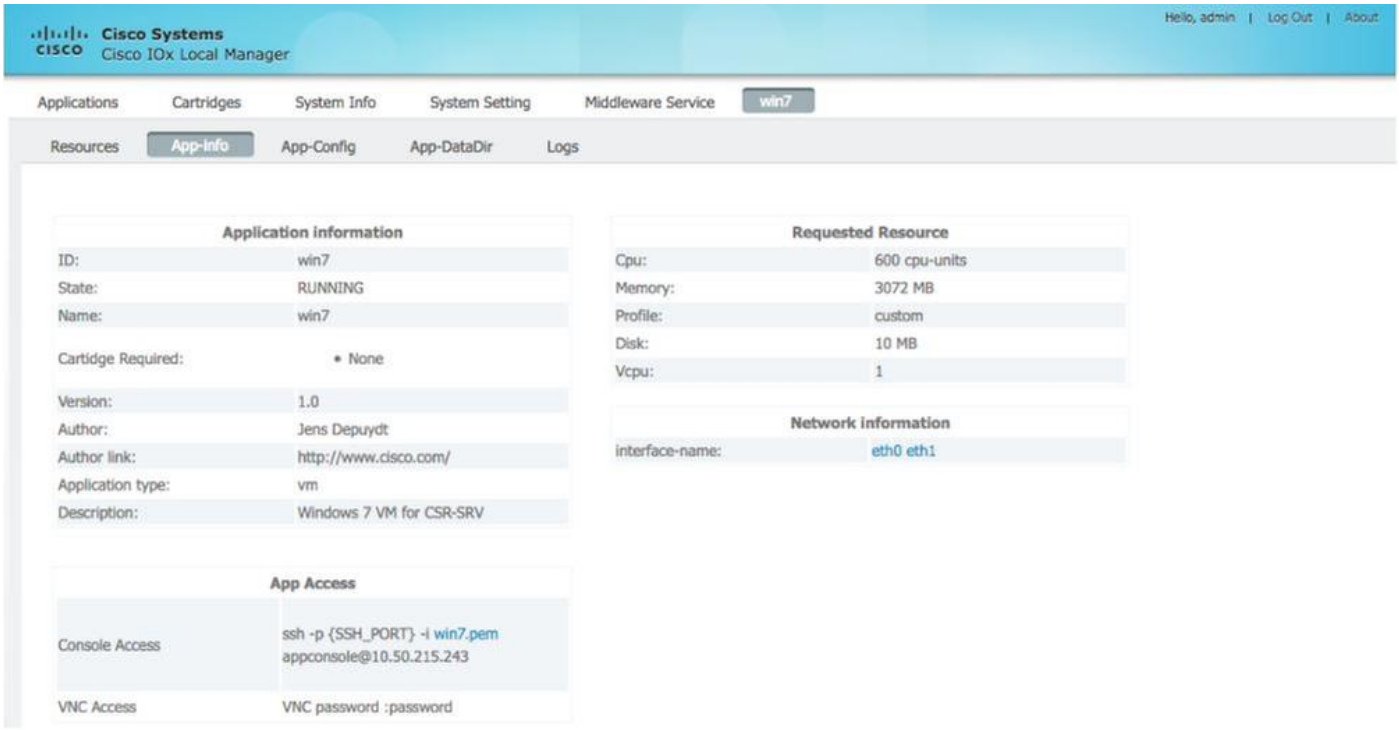
Verify

Use this section in order to confirm that your configuration works properly.

In order to check if the VM runs:

```
[root@cen7 ~]# ./ioxclient app list
Currently active profile :  CGR1120_20
Command Name:  application-list
Saving current configuration
List of installed App :
 1. win7        --->    RUNNING
```

You can also check the status from Local Manager as shown in the image:



Troubleshoot

This section provides information you can use in order to troubleshoot your configuration.

In order to troubleshoot issues with deployment, check the output of **ioxclient** or **/var/log/caf.log** on the CGM-SRV host OS.

Ensure that NAT is configured correctly to access all resources (Cisco application-hosting framework (CAF), Secure Shell (SSH), VNC).