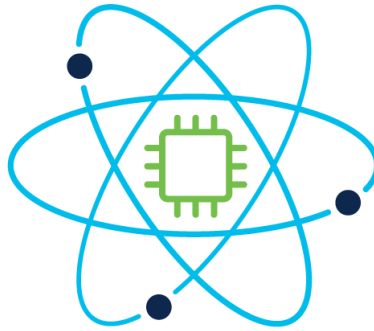


CML 2.5 Hands-on Lab



Powered by Cisco Modeling Labs

Cisco Modeling Labs (CML) is the go-to tool for the simulation of virtual Cisco network devices and beyond. In this workshop, we're going to cover the product from A to Z with a special focus on automation. In addition, we're going to show how to extend the platform by adding Kali Linux.

CML 2.5 Hands-on Lab

- Agenda

- Prerequisites

- Sections

 - Installation and licensing (demo)

 - Cockpit

 - Simulation life-cycle

 - Installation of third party image

 - Kali Image download and disk conversion

 - Steps to create a custom image

 - Create the node definition

 - Create the image definition

 - Create a new Kali node

 - Exploring and using the API via Swagger

 - Use of the Python Client Library (PCL)

vir1-utils: taking the PCL to the next level

Breakout tool

Terminal Server

Advanced usage

Appendix

Additional tools and links

Other custom images

Windows 10

Virtual WLC

vWLC Image Building

Running your own instance

Hardware

License

Software

VMware Workstation

Cisco Modeling Labs Software

Agenda

- Installation and licensing of the product (demo)
- Cockpit – system management, enabling SSH to access the Linux CLI
- Simulation Life cycle (simulation state, creating a topology, working with the UI)
- Installation of 3rd Party Image (node and image definition creation, image upload)
- Exploring and using the API via Swagger
- Use of the Python Client Library (PCL)
- vir1-utils: taking the PCL to the next level
- Breakout tool: what is it and how to use it
- Accessing consoles using the terminal server
- Additional references and closing

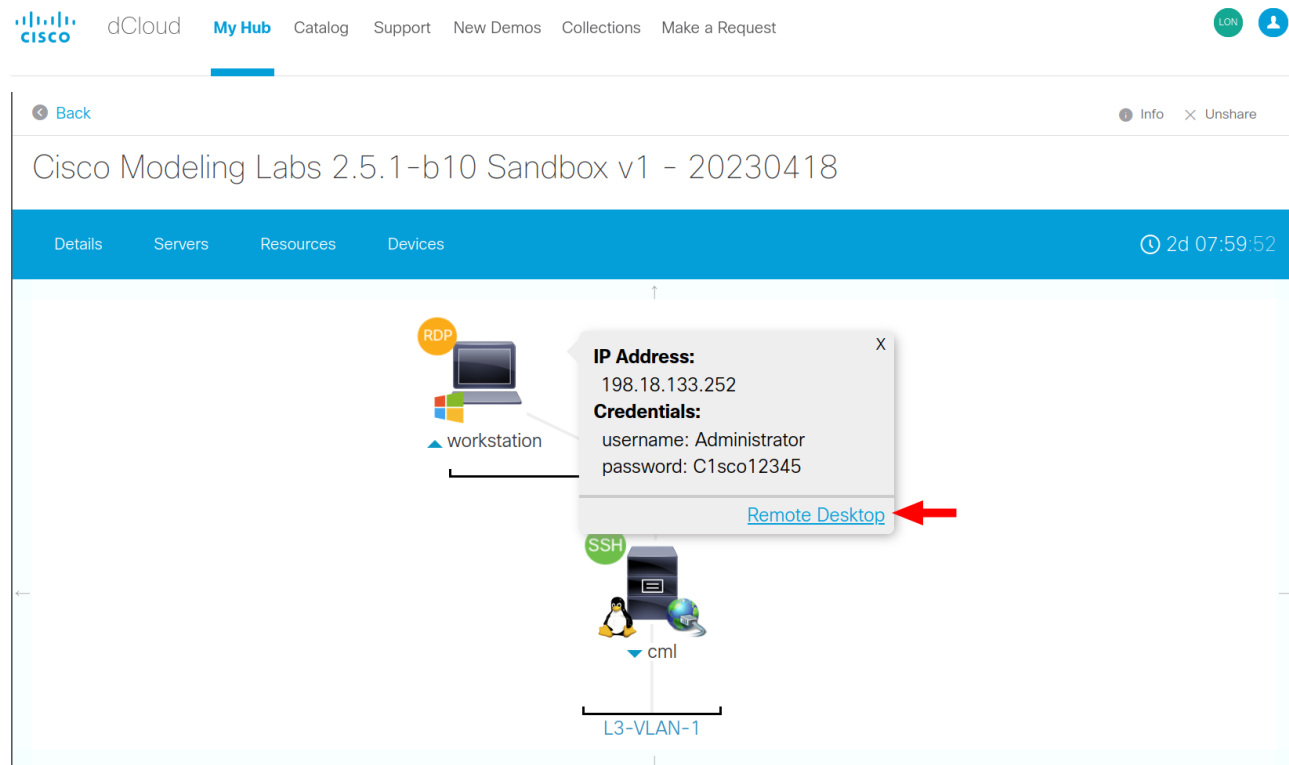
Prerequisites

The workshop will be running on Cisco dCloud. The installation section will be "demo-only", you can later refer to it in the recording.

A modern Web browser (Chrome / Firefox / Brave / Edge / Safari / ...) is required. There's a preference for Chrome(ish) browsers.

Important To access Cisco dCloud, each user must have a valid Cisco CCO ID. Otherwise, no access is possible.

A link will be provided during the WebEx session which will assign each attendee a dCloud lab (limited seats available. First come, first serve). When accessing this link, a page like this will be shown:



Click the "Remote Desktop" link for the workstation. A RDP session will open in your browser. It's recommended to take this RDP session to full screen and / or put it on a second display.

Refer to the appendix in case you want to repeat this workshop on your own hardware at home or at work.

Sections

Installation and licensing (demo)

Installation is done either as a virtual machine (preferred) or on a bare metal server. The VM requires an OVA to be deployed. For the bare metal install an ISO image is used to boot the server and install from there.

Once the OVA is deployed and the VM is started, the text based installer is launched. It will ask a few questions about user names, passwords, interfaces etc.

In addition, it is also determined whether CML should be deployed as an "all in one" or as a "cluster". Cluster deployments require two NICs per cluster member (controller and computes).

Part of the installation is also the copying of the so-called "reference platform ISO" with disk images to the hard drive of the controller. This is the part that takes the most time. Overall, the entire installation takes roughly 10-15 minutes.

Next, a license must be applied. This is done via the CML Web UI. The address is displayed on the screen of the machine when the installer has finished.

The licensing requires connectivity to the Internet to validate the license token. The token is generated via the software licensing center at <https://software.cisco.com>. It's a single string that can be copied and pasted into the Web UI. For enterprise licenses, additional node licenses can be configured, the standard license includes 20 Cisco node licenses.

Cockpit

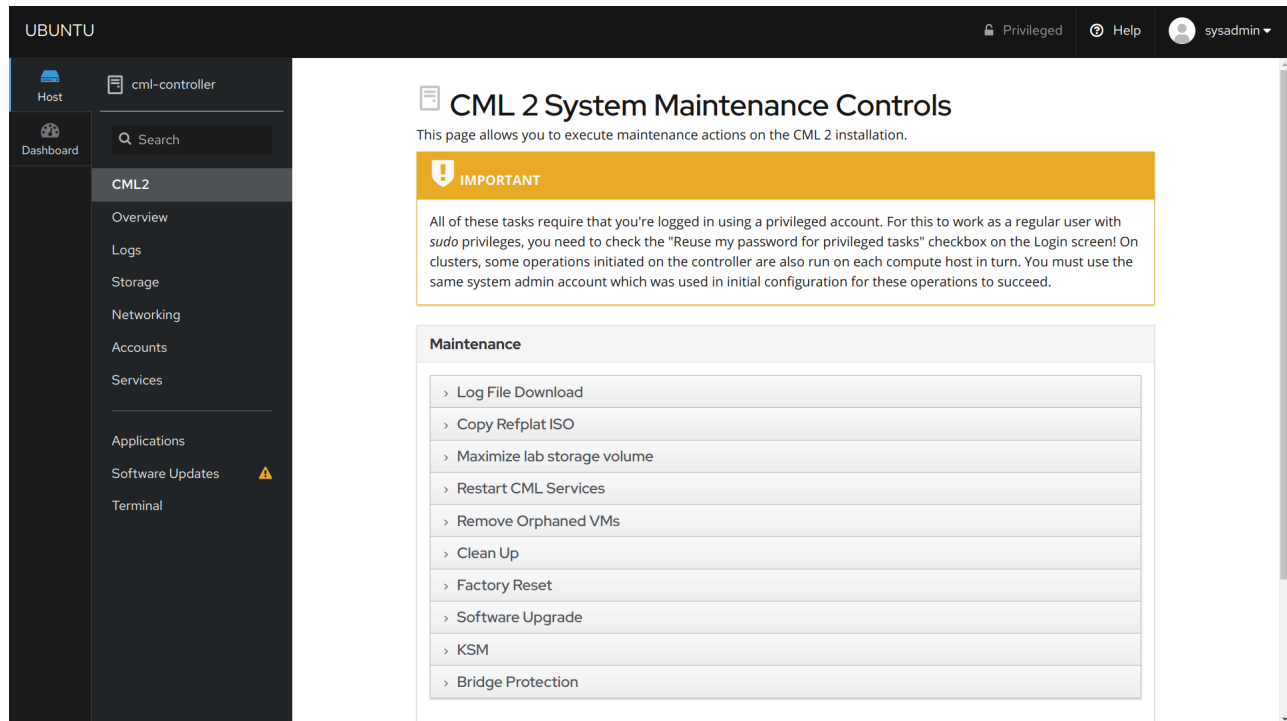
Certain system maintenance tasks are done via Cockpit, the built-in system admin tool. This includes:

- adding and configuring additional network interface cards (NICs)
- a command line terminal
- service management

In addition, a custom CML plugin allows for additional, CML specific tasks.

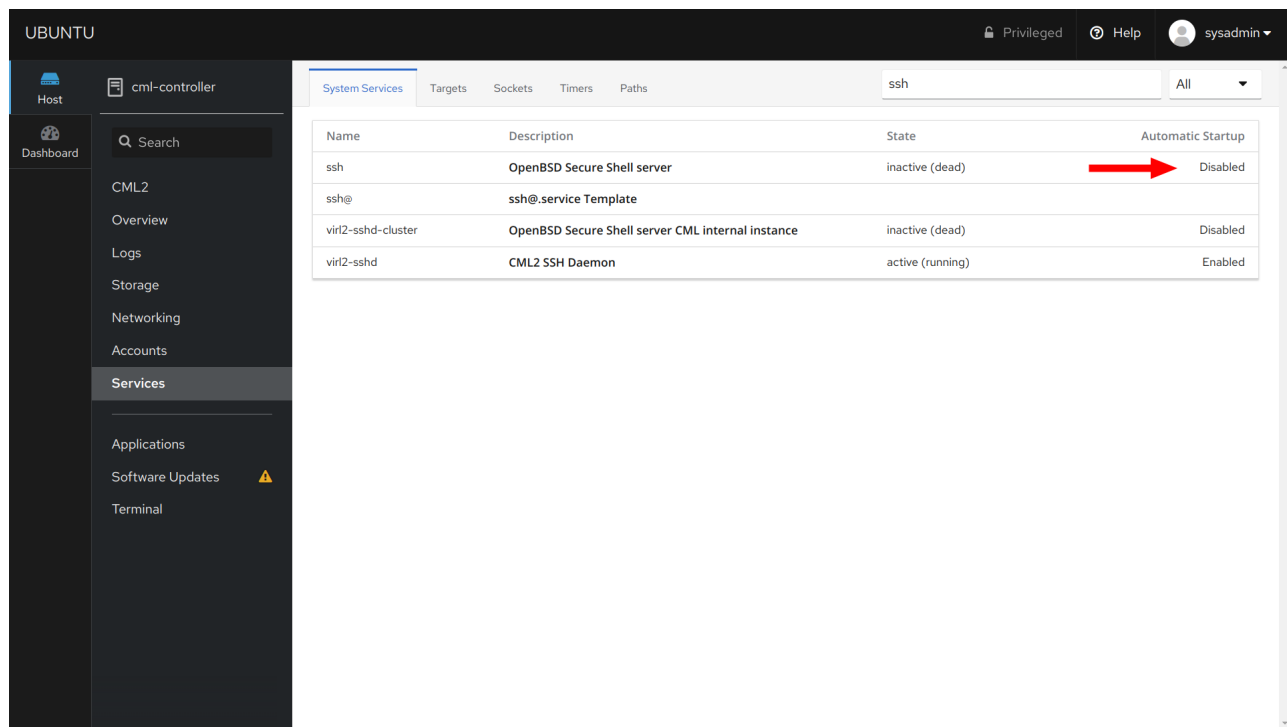
- CML maintenance tasks
- Upgrading CML software
- Restarting CML services
- etc.

After logging in, the CML specific section is shown:

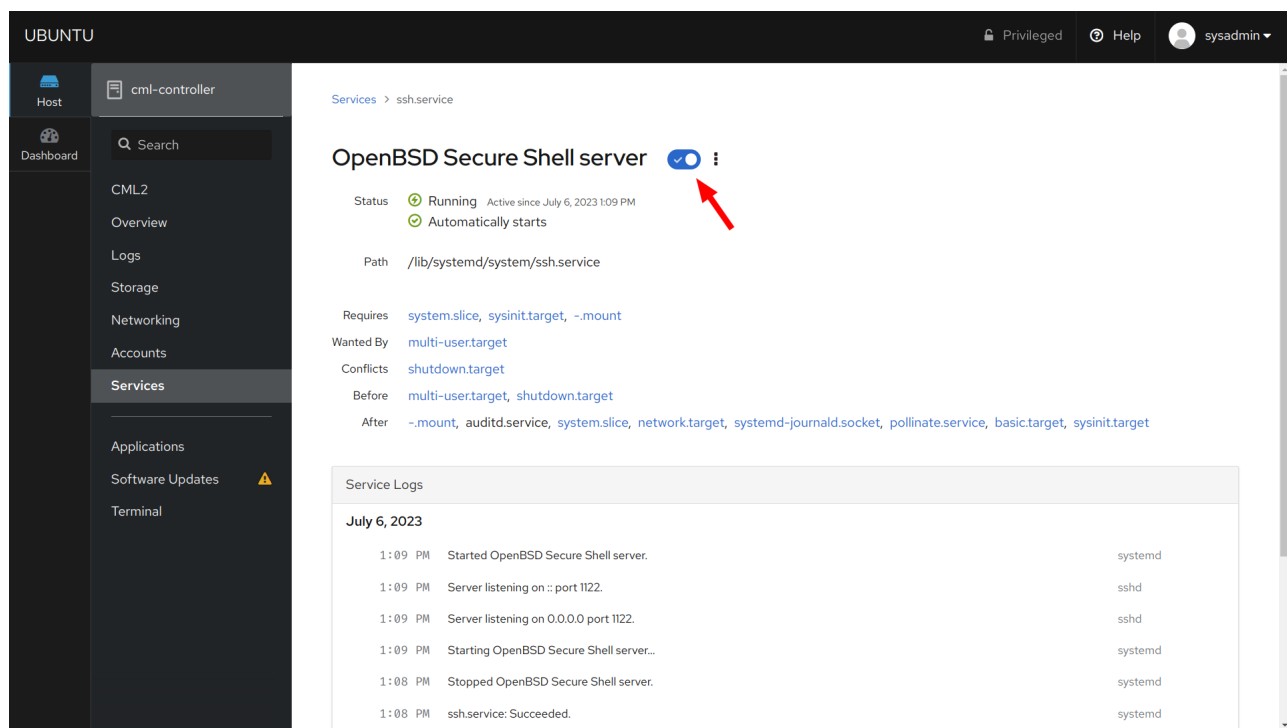


A common task done here is to re-enable the system SSH service which will then listen on port TCP/1122. It is disabled after installation as a security best-practice.

Go to "Services" and search for SSH:



Then click that table row, on the new page enable the service by clicking the "on" switch:



You should now be able to log into the CML host using SSH and get access to the command line:

```
sysadmin@cml: ~
C:\Users\Administrator>ssh -p1122 sysadmin@cml-controller
The authenticity of host '[cml-controller]:1122 ([198.18.134.1]:1122)' can't be established.
ECDSA key fingerprint is SHA256:srh+G3aQ0pCmyCRJeH4JFhXIpNJf+2arjM05+Q0zr0M.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[cml-controller]:1122' (ECDSA) to the list of known hosts.
Warning: the ECDSA host key for '[cml-controller]:1122' differs from the key for the IP address '[198.18.134.1]:1122'
Offending key for IP in C:\Users\Administrator/.ssh/known_hosts:2
Are you sure you want to continue connecting (yes/no)? yes
sysadmin@cml-controller's password:
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-147-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Thu Jul  6 14:41:26 UTC 2023

System load:  0.19           Processes:           356
Usage of /:   47.7% of 9.75GB Users logged in:      2
Memory usage: 2%            IPv4 address for bridge0: 198.18.134.1
Swap usage:   0%            IPv4 address for virbr0: 192.168.255.1

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge

156 updates can be applied immediately.
123 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Web console: https://cml:9090/

Last login: Thu Jul  6 14:16:38 2023 from 198.18.133.252
sysadmin@cml:~$
```

Note The dCloud CML variant has SSH enabled by default.

This concludes the Cockpit section.

Simulation life-cycle

This section simply teaches the user how to navigate the UI. The various sections to explore are:

- dashboard. where all labs are displayed either as tiles or in a list
- shared labs can be imported here by pressing the "Import" button
- new labs can be added by pressing the "Add" button
- actions on lab tiles include start, stop, wipe and delete

When adding a lab or opening an existing lab, the app opens the lab editor. This is the "heart" of the Web UI. Things to do here are

- editing a topology by adding nodes and links. Nodes can be dragged from the palette on the right and then dropped onto the canvas. Links can be added by right-clicking a node and selecting "Add link"

- configurations can be edited on "wiped" nodes
- nodes can be interacted with via the console or VNC tab (where available)
- on links, traffic conditions like delay or loss can be dialed in or a packet capture can be started
- annotations (ellipses, rectangles, lines, arrows, text) can be added and edited
- various display options can be toggled

Options / tabs on the bottom panel change depending on what kind of topology element is selected.

This concludes the Simulation life-cycle section.

Installation of third party image

We're going to install a Kali cloud image as a custom image into CML. The Kali image is ~170 MB in size and is provided as a cloud-image which can be installed in CML "as-is". The image is available [here](#).

Kali Image download and disk conversion

Execute these commands on the CML controller's command line by either using the Cockpit terminal or by using PuTTY to log into CML. Remember that the user name is `sysadmin` and the port to connect to is `1122`. The password is `C1sco012345`.

These commands don't require privileges, they can be run by the sysadmin user. We use `cURL` to download the image from the web.


```

$ curl -LO https://kali.download/cloud-images/kali-2023.2/kali-linux-
2023.2-cloud-genericcloud-amd64.tar.xz
  % Total    % Received % Xferd  Average Speed   Time    Time
Time  Current
                                 Dload  Upload  Total  Spent
Left  Speed
100 173M 100 173M    0     0 4634k      0  0:00:38  0:00:38 --:--
-:-- 4984k
$ tar xvf kali-linux-2023.2-cloud-genericcloud-amd64.tar.xz
disk.raw
$ ls -lh
total 1.5G
-rw-r--r--. 1 rschmied rschmied 12G May 23 05:13 disk.raw
-rw-rw-r--. 1 rschmied rschmied 174M Jul  5 19:17 kali-linux-2023.2-
cloud-genericcloud-amd64.tar.xz
$

```

Next, we convert the "raw" disk to a QCOW2 disk. The flags ensure progress is shown and that the result is compressed. We then remove the intermediate files to free up space. Finally, we print some information about the created disk image.

```

$ qemu-img convert disk.raw -c -p -O qcow2 kali.qcow2
(100.00/100%)
$ rm kali-*.tar.xz disk.raw
$ qemu-img info kali.qcow2
image: kali.qcow2
file format: qcow2
virtual size: 12 GiB (12884901888 bytes)
disk size: 294 MiB
cluster_size: 65536
Format specific information:
  compat: 1.1
  compression type: zlib
  lazy refcounts: false
  refcount bits: 16
  corrupt: false
  extended l2: false
$

```

This file is now moved into the "dropfolder". The dropfolder is the location where the CML controller looks for new images. This is the same location where files will show up when uploaded through the UI or via SCP on port TCP/22 to the controller.

```
$ sudo cp kali.qcow2 /var/local/virl2/dropfolder
$ rm kali.qcow2
```

We have to copy and can not move the file as the home directory and the VIRL2 directory usually are on different file systems. On the dCloud instance, `mv` works as well.

Steps to create a custom image

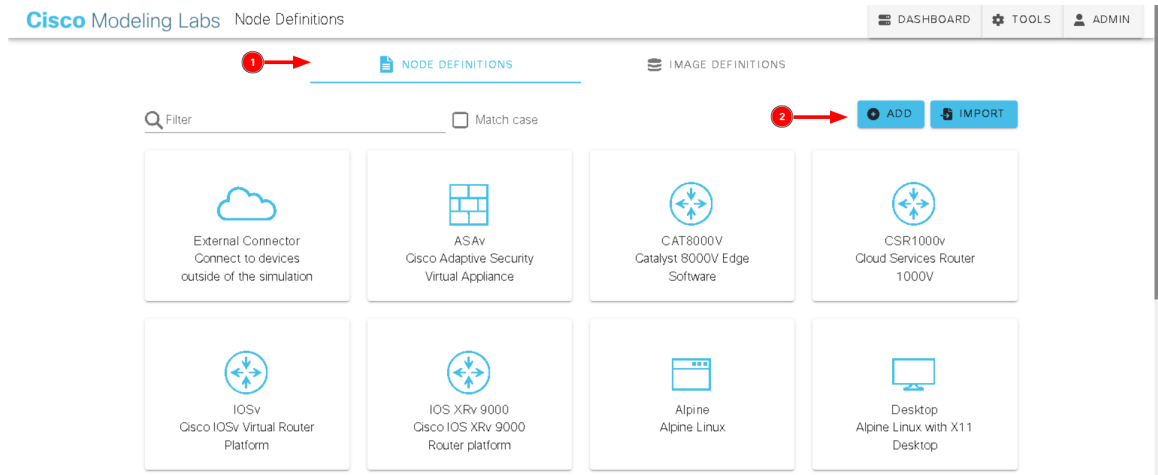
To make the image we've just created available in CML, we need to go through a couple of steps:

- Upload the disk image to the controller. This can be done either via the UI or via SCP. For larger images, the upload via SCP is recommended (this is already done as part of the previous steps)
- Create a node definition for the new node type
- Create an image definition for the uploaded image
- Link the image definition to the node definition by referencing the node definition ID within the image definition

Create the node definition

- Navigate to *Tools* ➔ *Node and Image Definitions*
- Ensure the "Node definition" tab is selected

- Click the "Add" button



Filling the form is a bit tedious, so here's a quick summary what should go into the fields:

- ID should be "kali"
- Description whatever you want
- Nature is "server"
- Image definition is left blank for the moment
- User interface
 - "visible" is on
 - Description is whatever you want
 - prefix should be "kali-" (with the dash)
 - Icon is "server"
 - Label is "kali"
- Linux Native Simulation
 - Domain driver is "KVM"
 - Disk Driver is "virtio"
 - Set Memory to 512
 - CPUs to 1
 - CPU limit to 100
 - Network driver is "virtio"
 - Boot Disk size is 16 (GB). Make sure that the size here is at least as big as the size reported by the Qemu image tool (see above, the virtual size was reported as 12 GB)
 - Video model is "Cirrus"

- Video memory us set to "64"
- Interfaces
 - "Loopback interface" is set to off
 - serial ports is set to 1
 - default number of physical interfaces is set to 1
 - add four interfaces and name them eth0, eth1, eth2 and eth3
- Boot
 - set timeout to 120
 - no specific boot line required, defaults should be fine
- pyATS is disabled
- Property inheritance is unchanged (all on)
- Configuration generator is deselected (can't generate configurations for Kali)
- Provisioning is "off"

Then click the "Create" button -- if anything is missing then the form will report it, otherwise things can still be changed after creation.

Create the image definition

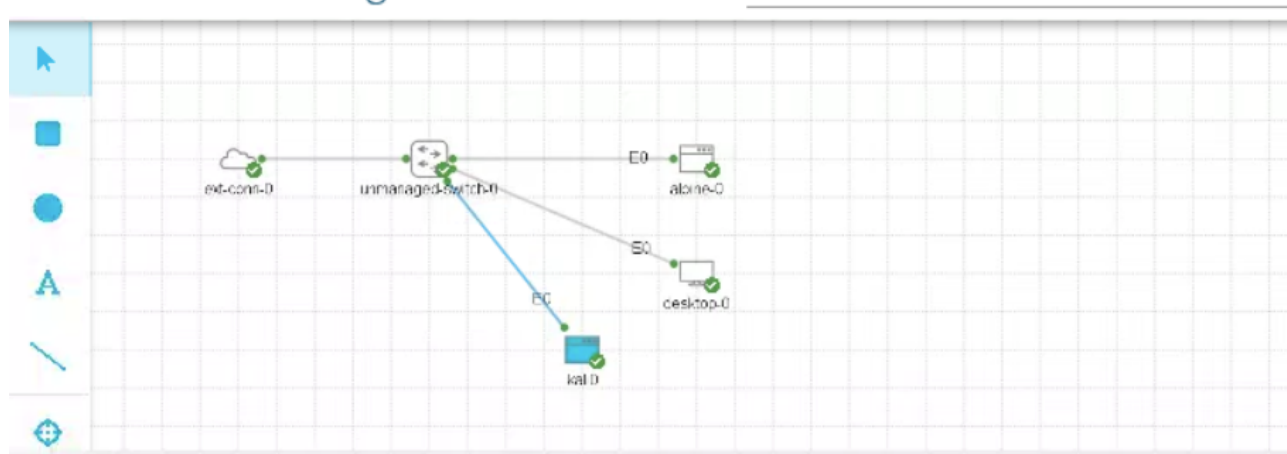
- Navigate to *Tools ➔ Node and Image Definitions*
- Ensure the "Image definition" tab is selected
- Click the "Add" button
- ID is "kali-6-1-0" or similar (needs to be unique and should consider the fact that a newer image version might be added in the future)
- Label is "Kali 6.1.0"
- Description is whatever you like
- Select the disk image from the drop down list that we put into the drop folder
`kali.qcow2`
- Select the associated "kali" Node definition from the drop down list

Leave the rest as-is and click the "Create" button. This finalizes the new Kali node and image definition.

Create a new Kali node

We can now either create a new Kali lab or add a Kali node to an existing lab. After linking it to an external connector via an unmanaged switch it will happily boot and acquire an IP address (ensure the external connector configuration uses the NAT bridge):

Cisco Modeling Labs Workbench Lab at Thu 14:18 PM



The network diagram shows a cloud node connected to an unmanaged switch. The switch is connected to a desktop node and a Kali node. The desktop node is connected to a switch, which is connected to a desktop node. The Kali node is connected to a switch, which is connected to a desktop node.

NODE INFO **SIMULATE** **CONNECTIVITY** **CONSOLE** **VNC** **EDIT CONFIG**

```
The programs included with the Kali GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Thu Jul 6 15:55:36 UTC 2023 on tty1  
(Message from Kali developers)  
  
This is a minimal installation of Kali Linux, you likely  
want to install supplementary tools. Learn how:  
⇒ https://www.kali.org/docs/troubleshooting/common-minimum-setup/  
  
This is a cloud installation of Kali Linux. Learn more about  
the specificities of the various cloud images:  
⇒ https://www.kali.org/docs/troubleshooting/common-cloud-setup/  
  
(Run: "touch ~/.hushlogin" to hide this message)  
root@kali:~#
```

CPU 0.94% MEMORY 2.88%

This was especially easy since Kali automatically provides a console on the serial port (and on the graphical screen via VNC since we provided a graphics adapter which isn't strictly necessary).

For other operating systems, some experimentation is required...

- what disk driver is required?

- what NIC types are supported?
- do we need a graphics adapter or not? Which one? How much memory?
- Is additional storage / a data disk required?
- what needs to be done to get a serial console (if any at all)
- and potentially more...

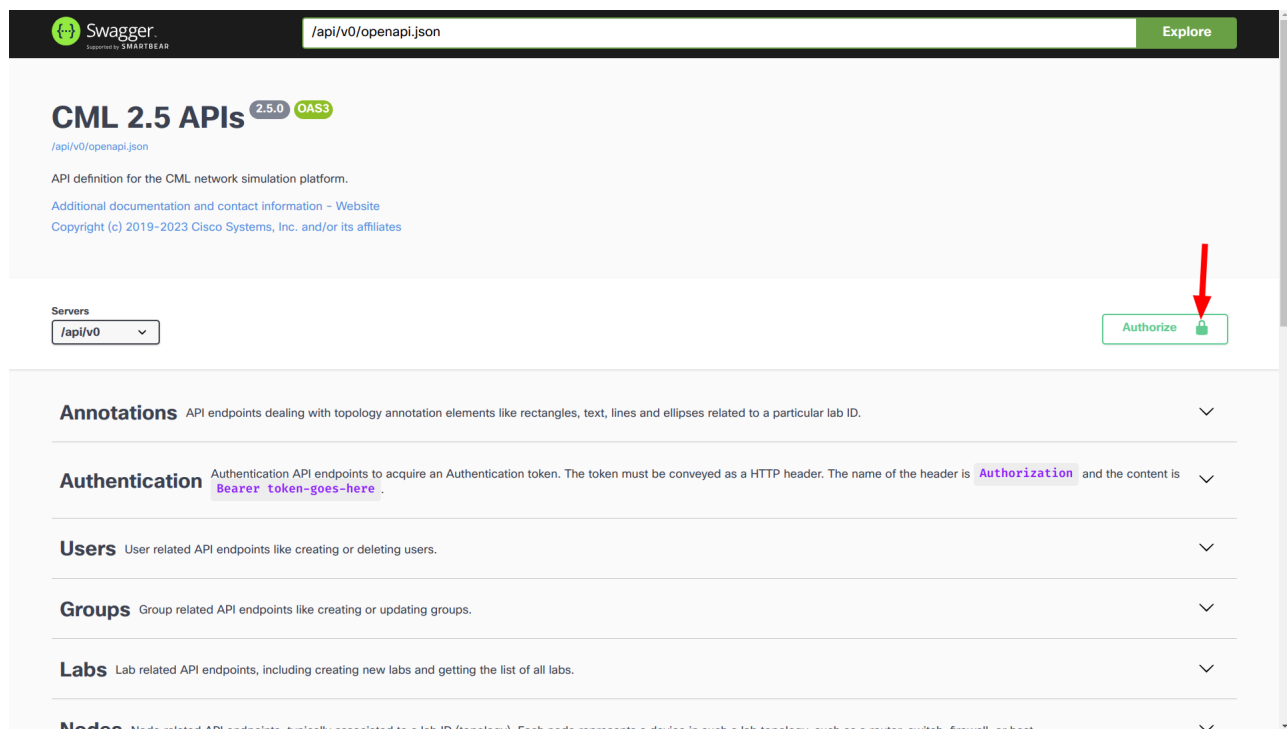
In some cases (like with Windows 11) some virtual hardware might be needed which isn't currently available on CML (in this case, the TPM).

This concludes the third party image section.

Exploring and using the API via Swagger

CML includes complete API documentation. It has a convenient front end called "Swagger" which not only shows all the details of the API including data schema but also allows to run API calls and immediately see the results, typically in JSON.

From the CML UI, navigate to *Tools* → *API Documentation*. Note that the pad lock is closed which means that the API browser is already "authorized". If the lock is open then the API browser isn't authorized and a proper token must be provided using the authentication API endpoint.



The screenshot displays the Swagger UI for the CML 2.5 APIs. At the top, the Swagger logo is visible next to the API definition file path `/api/v0/openapi.json`. Below this, the title "CML 2.5 APIs" is shown with version tags "2.5.0" and "OAS3". The main content area lists several API endpoint categories, each with a brief description and a dropdown arrow for further details:

- Annotations**: API endpoints dealing with topology annotation elements like rectangles, text, lines and ellipses related to a particular lab ID.
- Authentication**: Authentication API endpoints to acquire an Authentication token. The token must be conveyed as a HTTP header. The name of the header is `Authorization` and the content is `Bearer token-goes-here`.
- Users**: User related API endpoints like creating or deleting users.
- Groups**: Group related API endpoints like creating or updating groups.
- Labs**: Lab related API endpoints, including creating new labs and getting the list of all labs.
- Nodes**: Node related API endpoints, typically associated to a lab ID (hostname). Each node represents a device in such a lab topology, such as a router, switch, firewall, or host.

In the top right corner, there is an "Authorize" button with a closed padlock icon, indicating that the API browser is already authorized. A red arrow points to this button.

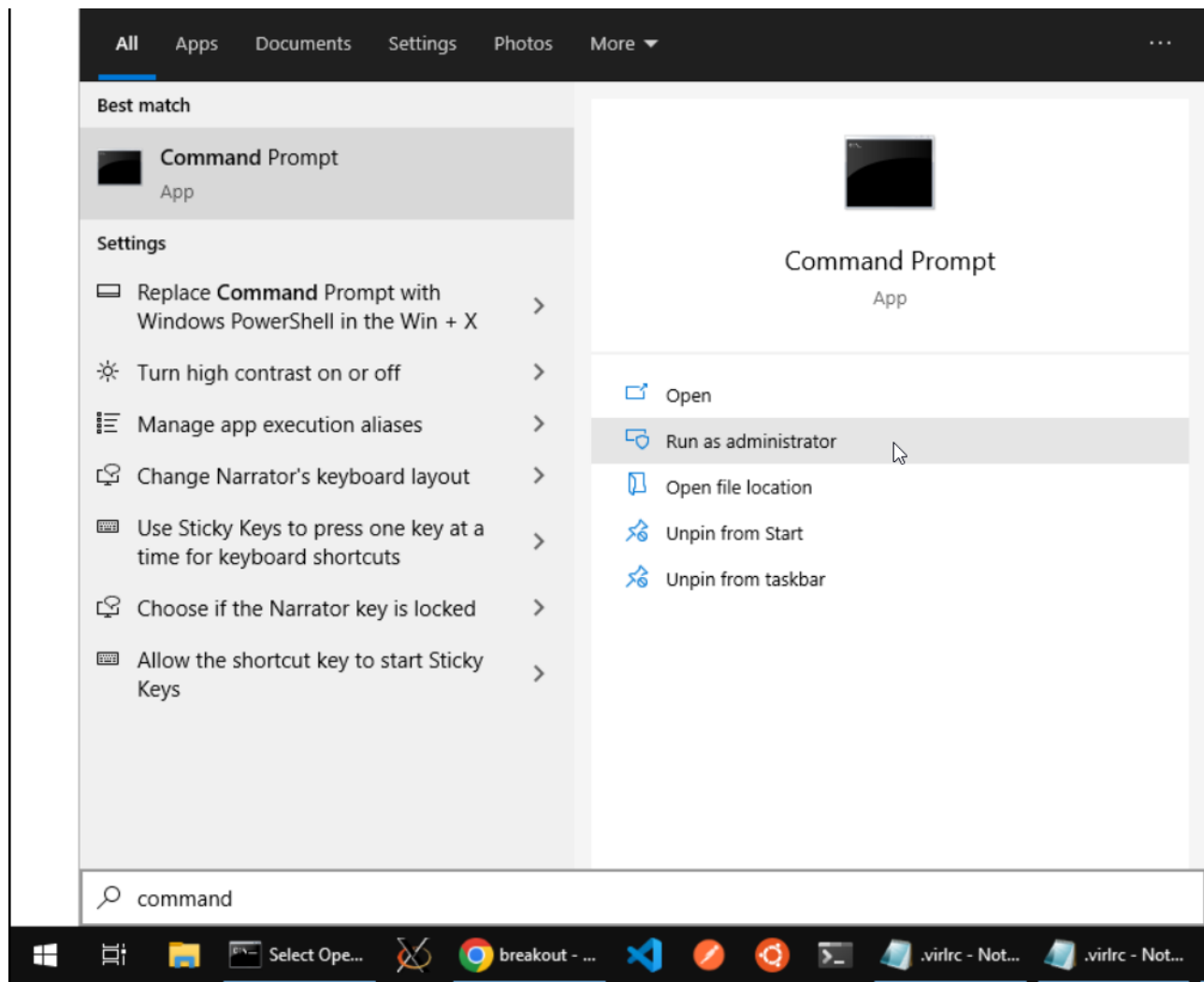
Experiment with various API calls like:

- acquire an authentication token
- list all labs
- list nodes of a lab
- start or stop a lab
- check system health
- list known node definitions

This concludes the API via Swagger section.

Use of the Python Client Library (PCL)

The PCL is available for download from the controller or on PyPI (Python package index). On the dCloud demo instance, the package is already installed. For the following sections we need a Terminal / CLI access. On the Windows machine, click Start, then type "command" (for "Command prompt") and **run it as administrator**:



The *administrator* bit is important to make some commands work later on. Note that this will start the shell in the Windows system directory. Change the working directory to the Administrator's home directory using:

```
cd C:\Users\Administrator
```

We'll check now via `pip list` what Python packages are already installed. In this case, the `vir12-client` (Python Client Library / PCL) is already there but it is outdated (see the output below). We're going to upgrade it to the latest version via `pip install --upgrade vir12-client`:


```
Command Prompt
certifi      2020.4.5.1
chardet      3.0.4
charset-normalizer 2.0.12
click        7.1.1
cmlutils     1.2.1
docopt       0.6.2
Flask        1.1.2
gunicorn     20.0.4
idna         2.9
itsdangerous 1.1.0
Jinja2       2.11.1
lxml         4.5.0
MarkupSafe   1.1.1
pip          22.1.2
python-dateutil 2.8.1
PyYAML       5.3.1
requests     2.28.0
requests-toolbelt 0.9.1
setuptools   41.2.0
six          1.14.0
tabulate     0.8.7
urllib3      1.26.9
virl2-client 2.4.0+build.3
werkzeug     1.0.1
wheel        0.35.1

[notice] A new release of pip available: 22.1.2 -> 23.1.2
[notice] To update, run: python.exe -m pip install --upgrade pip
C:\Users\Administrator>pip install --upgrade virl2-client
```

If the PCL is not installed, then a simple `pip install virl2-client` will do.

***Note** the latest dCloud CML environment has the correct version already installed.*

The documentation for the PCL is built into the CML controller and can be opened by navigating to *Tools* ➔ *Client Library* which opens a new tab.

The screenshot shows a web browser window displaying the 'virl2_client' documentation. The page has a dark sidebar with a search bar and a table of contents. The main content area is titled 'CML 2 API Client Documentation' and includes a description of the client library, a 'Content' section with links to 'Introduction', 'Examples', and 'virl2_client package', and a list of examples. The examples list includes 'Creating a lab with nodes and links', 'Stopping all the labs', 'Getting all lab names', 'Stopping all labs of a User', 'Uploading an image disk file', 'Using the Client Library with Netmiko', 'Licensing the System', and 'Using Link Conditioning'. The 'virl2_client package' section lists 'InitializationError' and 'InterfaceNotFound'.

The documentation has a couple of examples to work through:

- create a lab with nodes and links
- stopping all labs
- getting all lab names

You can copy and paste these examples into a file (via Notepad, for example) and run them via the command line like this:

```
python demo1.py
```

This example assumes you've copied the code example into a file called `demo1.py`.

Ensure that you provide a resolvable hostname and disable the SSL verification. Here's a working example:

```
from vir12_client import ClientLibrary
client = ClientLibrary("https://cml", "admin", "Cisco12345",
ssl_verify=False)

all_lab_names = [lab.title for lab in client.all_labs()]
print(all_lab_names)
```

```
C:\Users\Administrator>more demo.py
from vir12_client import ClientLibrary
client = ClientLibrary("https://cml", "admin", "Cisco12345", ssl_verify=False)

all_lab_names = [lab.title for lab in client.all_labs()]
print(all_lab_names)

C:\Users\Administrator>python demo.py
SSL Verification disabled
['Lab at Thu 14:18 PM']

C:\Users\Administrator>_
```

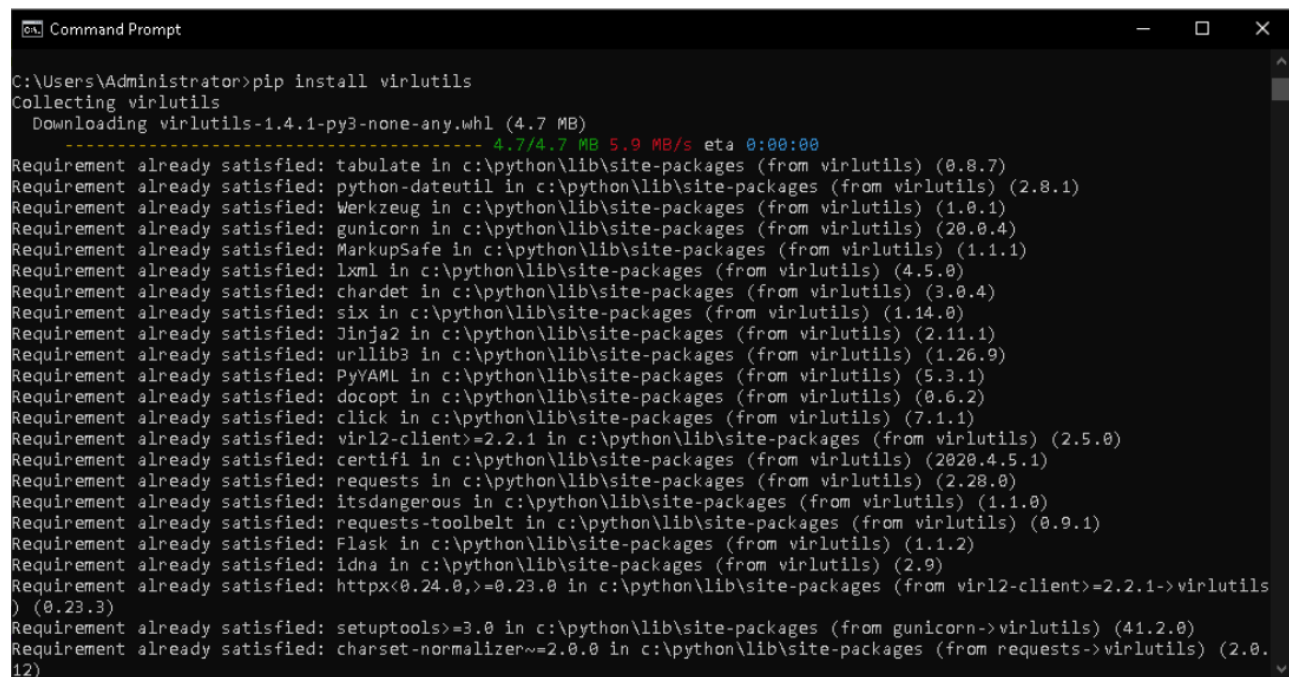
This concludes the PCL section.

virl-utils: taking the PCL to the next level

virl-utils is a tool written in Python and available on GitHub and PyPI. It is inspired by Vagrant tool from Hashicorp and uses the PCL extensively. In fact, since it's available on GitHub, you can study it as a good example how to use the PCL.

The basic idea is to put a topology file and a configuration file into a directory and then do a `virl up` to start the lab. After testing, a `virl down` will stop the lab again.

virl-utils can be easily installed via pip. At the command prompt run `pip install virlutils` (note there's no dash!), it might pull in a couple of dependencies:



```
Command Prompt
C:\Users\Administrator>pip install virlutils
Collecting virlutils
  Downloading virlutils-1.4.1-py3-none-any.whl (4.7 MB)
    ----- 4.7/4.7 MB 5.9 MB/s eta 0:00:00
Requirement already satisfied: tabulate in c:\python\lib\site-packages (from virlutils) (0.8.7)
Requirement already satisfied: python-dateutil in c:\python\lib\site-packages (from virlutils) (2.8.1)
Requirement already satisfied: Werkzeug in c:\python\lib\site-packages (from virlutils) (1.0.1)
Requirement already satisfied: gunicorn in c:\python\lib\site-packages (from virlutils) (20.0.4)
Requirement already satisfied: MarkupSafe in c:\python\lib\site-packages (from virlutils) (1.1.1)
Requirement already satisfied: lxml in c:\python\lib\site-packages (from virlutils) (4.5.0)
Requirement already satisfied: chardet in c:\python\lib\site-packages (from virlutils) (3.0.4)
Requirement already satisfied: six in c:\python\lib\site-packages (from virlutils) (1.14.0)
Requirement already satisfied: Jinja2 in c:\python\lib\site-packages (from virlutils) (2.11.1)
Requirement already satisfied: urllib3 in c:\python\lib\site-packages (from virlutils) (1.26.9)
Requirement already satisfied: PyYAML in c:\python\lib\site-packages (from virlutils) (5.3.1)
Requirement already satisfied: docopt in c:\python\lib\site-packages (from virlutils) (0.6.2)
Requirement already satisfied: click in c:\python\lib\site-packages (from virlutils) (7.1.1)
Requirement already satisfied: virl2-client>=2.2.1 in c:\python\lib\site-packages (from virlutils) (2.5.0)
Requirement already satisfied: certifi in c:\python\lib\site-packages (from virlutils) (2020.4.5.1)
Requirement already satisfied: requests in c:\python\lib\site-packages (from virlutils) (2.28.0)
Requirement already satisfied: itsdangerous in c:\python\lib\site-packages (from virlutils) (1.1.0)
Requirement already satisfied: requests-toolbelt in c:\python\lib\site-packages (from virlutils) (0.9.1)
Requirement already satisfied: Flask in c:\python\lib\site-packages (from virlutils) (1.1.2)
Requirement already satisfied: idna in c:\python\lib\site-packages (from virlutils) (2.9)
Requirement already satisfied: httpx<0.24.0,>=0.23.0 in c:\python\lib\site-packages (from virl2-client>=2.2.1->virlutils) (0.23.3)
Requirement already satisfied: setuptools>=3.0 in c:\python\lib\site-packages (from gunicorn->virlutils) (41.2.0)
Requirement already satisfied: charset-normalizer<=2.0.0 in c:\python\lib\site-packages (from requests->virlutils) (2.0.12)
```

After the installation has succeeded, the `virl` command is available:

```
Command Prompt
C:\Users\Administrator>virl
Usage: virl [OPTIONS] COMMAND [ARGS]...

Options:
  --debug / --no-debug  Print any debugging output.
  --help                Show this message and exit.

Commands:
  clear      clear the current lab ID
  cockpit    opens the Cockpit UI
  command    send a command or config to a node (requires pyATS)
  console    console for node
  definitions manage image and node definitions
  down       stop a lab
  extract    extract configurations from all nodes in a lab
  generate   generate inv file for various tools
  id         get the current lab title and ID
  license    work with product licensing
  ls         lists running labs and optionally those in the cache
  nodes     get node list for the current lab
  pull      pull topology.yaml from repo
  rm        remove a lab
  save      save lab to a local yaml file
  search    list topologies available via github
  ssh       ssh to a node
  start     start a node
  stop      stop a node
  telnet    telnet to a node
  ui        opens the Workbench for the current lab
  up        start a lab
  use       use lab launched elsewhere
  version   version information
  wipe     wipe a lab or nodes within a lab

C:\Users\Administrator>
```

After installation, the tool must be configured, the documentation is [available here](#). For this demo, we simply copy and change the user name from `demo` to `admin` in the `.virlrc` file that was already there. Also ensure that the `VIRL_HOST` is set to `cm1`. With a command prompt open in the Administrator's home directory `C:\Users\Administrator`, the file can be edited by running `notepad .virlrc`:

```
.virlrc - Notepad
File Edit Format View Help
VIRL_HOST=cm1
VIRL_USERNAME=admin
VIRL_PASSWORD=Cisco12345
#CML_VERIFY_CERT=c:\ca_bundle.pem
CML_VERIFY_CERT=false
CML2_PLUS="yes"
```

Don't forget to save the file!

```
Command Prompt
extract    extract configurations from all nodes in a lab
generate  generate inv file for various tools
id         get the current lab title and ID
license   work with product licensing
ls         lists running labs and optionally those in the cache
nodes     get node list for the current lab
pull      pull topology.yaml from repo
rm         remove a lab
save      save lab to a local yaml file
search    list topologies available via github
ssh       ssh to a node
start     start a node
stop      stop a node
telnet    telnet to a node
ui        opens the Workbench for the current lab
up        start a lab
use       use lab launched elsewhere
version   version information
wipe      wipe a lab or nodes within a lab

C:\Users\Administrator>notepad .virlrc
C:\Users\Administrator>virl version
virlutils Version: 1.4.1
CML Controller Version: 2.5.1+build.10

C:\Users\Administrator>virl ls
Labs on Server
```

ID	Title	Description	Owner	Status	Nodes	Links	Interfaces
987cd265-92d1-4d05-a8ea-a6fe3f067e99	test		00000000-0000-4000-a000-000000000000	DEFINED_ON_CORE	2	1	5

```
C:\Users\Administrator>
```

After the tool is configured OK, we can use the various sub commands of the `virl` command like

- list labs via `ls`
- open consoles into nodes
- start / stop / wipe / remove a lab

The tool uses the concept of a "current lab". The current lab is automatically set when starting it via `virl up`. You might have started a lab already using the UI or via previous exercises. In this case you can do

```
$ virl use -n test
```

Assuming your lab is called test (as in the screen shot above). From here, you can now work with the nodes in the lab:

```
OpenSSH SSH client

c:\Users\Administrator>virl nodes
No current lab selected

c:\Users\Administrator>virl use -n test

c:\Users\Administrator>virl nodes
Here is a list of nodes in this lab
```

ID	Label	Type	Compute Node	State	Wiped?
11ff4108-8db4-4fae-b9dd-9b481e4e2228	alpine-0	alpine	cml	BOOTED	False
f2bd19a0-4c65-4b4f-9271-8476d967e7e0	ext-conn-0	external_connector	cml	BOOTED	False

```


c:\Users\Administrator>virl console alpine-0
The authenticity of host 'cml.demo.dcloud.cisco.com (198.18.134.1)' can't be established.
RSA key fingerprint is SHA256:bkC+kkbWw1u+IQNWZsjfXu1qRRkUyG94gw2lqSJLKvQ.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'cml.demo.dcloud.cisco.com,198.18.134.1' (RSA) to the list of known hosts.
admin@cml.demo.dcloud.cisco.com's password:
Connecting to console for alpine-0
Connected to terminalserver.
Escape character is '^]'.

Welcome to Alpine Linux 3.16
Kernel 5.15.68-0-virt on an x86_64 (/dev/ttyS0)

inserthostname-here login: _
```

This concludes the virl-utils section.

Breakout tool

The breakout tool allows to use native tools like PuTTY or VNC clients to connect to node consoles or graphical user interfaces.

It is available on the controller to download for Windows, Linux and macOS (Apple Silicon and Intel variants available).

The breakout tool is already installed on the dCloud Windows machine, but it might not be the current version. We can download the current version from the controller by navigating to *Tools* ➔ *Breakout Tool*. Then download the Windows version:

CML2 Breakout

Search docs

CONTENT

General Usage

Download

Breakout Usage via GUI

Use of Breakout on macOS

Windows Shortcut Creation

Protocol Handlers

Appendix

/ Download

[View page source](#)

Download

The Breakout tool is a small (~12MB) command line utility that is available for Linux, macOS and Windows.

When downloading, it may need to be made "executable" or exempt from the security mechanics on the operating system so that it can be executed. Go here [Use of Breakout on macOS](#) for specific instructions on macOS after downloading.

All versions are 64 bit versions.

Operating System	Download link
Linux Version (Intel)	breakout-linux-amd64
Linux Version (ARM)	breakout-linux-arm64
macOS Version (Intel)	breakout-darwin-amd64
macOS Version (Apple Silicon)	breakout-darwin-arm64
Windows Version (64-bit x86)	breakout-windows-amd64.exe

Note

The above download links *only* work when the documentation is viewed on the CML² controller.

Previous

Next

© Copyright Copyright (c) 2019-2023, Cisco Systems, Inc..

Built with [Sphinx](#) using a [theme](#) provided by [Read the Docs](#).

Copy the file into the current directory, renaming it to `breakout.exe`:

```
C:\Users\Administrator>dir Downloads
Volume in drive C has no label.
Volume Serial Number is 3E2B-2F1C

Directory of C:\Users\Administrator\Downloads

07/06/2023  12:04 AM    <DIR>          .
07/06/2023  12:04 AM    <DIR>          ..
07/06/2023  12:04 AM             12,962,304 breakout-windows-amd64.exe
               1 File(s)          12,962,304 bytes
               2 Dir(s)  68,147,269,632 bytes free

C:\Users\Administrator>copy Downloads\breakout-windows-amd64.exe breakout.exe
Overwrite breakout.exe? (Yes/No/All): yes
        1 file(s) copied.

C:\Users\Administrator>
```

Next, we need to verify the configuration. If nothing has been pre-configured then the steps are:

- run "config" -- `breakout config` to create a template configuration file
- edit the configuration file to match your controller address, username and password using `notepad config.yaml`, don't forget to save the file. The controller address is `https://cml` in dCloud. **Ensure that there is no trailing slash!** Change the listen address to `localhost`. It should look like this:

```
console_start_port: 9000
controller: https://cml
extra_lf: false
lab_config_name: labs.yaml
listen_address: localhost
password: C1sco12345
populate_all: false
ui_server_port: 8080
username: admin
verify_tls: false
vnc_start_port: 5900
```

- start a lab (in case none is running at the moment). To use VNC, have at least one Desktop node running in a lab
- run "init" -- `breakout init` to download information about running labs from the controller

- edit the created `labs.yaml` file and set the `enabled` flag to `true` for the labs you want to use. Note that if multiple labs are enabled, you need to ensure that there's no conflict in the ports being used.

 labs.yaml - Notepad

File Edit Format View Help

```
987cd265-92d1-4d05-a8ea-a6fe3f067e99:
```

```
  created: "2023-07-06T04:51:26+00:00"
```

```
  enabled: true|  set to true from false
```

```
  id: 987cd265-92d1-4d05-a8ea-a6fe3f067e99
```

```
  lab_description: ""
```

```
  lab_title: test
```

```
  link_count: 1
```

```
  node_count: 2
```

```
  nodes:
```

```
    11ff4108-8db4-4fae-b9dd-9b481e4e2228:
```

```
      devices:
```

```
        - enabled: true
```

```
          listen_port: 9000
```

```
          name: serial0
```

```
          running: false
```

```
          status: ""
```

```
        - enabled: true
```

```
          listen_port: 5900
```

```
          name: vnc
```

```
          running: false
```

```
          status: ""
```

```
      label: alpine-0
```

```
state: STARTED
```

- run "run" -- `breakout run` to actually run the tool and make the ports available

```
C:\Users\Administrator>vi labs.yaml
'vi' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Administrator>notepad labs.yaml

C:\Users\Administrator>breakout.exe run
System version: 2.5.1+build.10
+-----+-----+-----+
| NODE LABEL | DEVICE | ADDRESS |
+-----+-----+-----+
| alpine-0   | serial0 | TCP/9000 |
+-----+-----+-----+
|            | vnc     | TCP/5900 |
+-----+-----+-----+

Running... Press Ctrl-C to stop.
```

- use your native clients to connect to the ports displayed, there's PuTTY and UltraVNC available

Alternatively, most of these steps can also be done via the convenient "breakout UI" which is available when running `breakout ui`. When running the UI, point your browser to the address displayed in the command prompt:

```
Running... Serving UI/API on http://localhost:8080, Ctrl-C to stop.
```

The "listening address" and port (here localhost and 8080) can be changed in the `config.yaml` or in the UI itself.

Note changes to those parameters require a restart of the tool

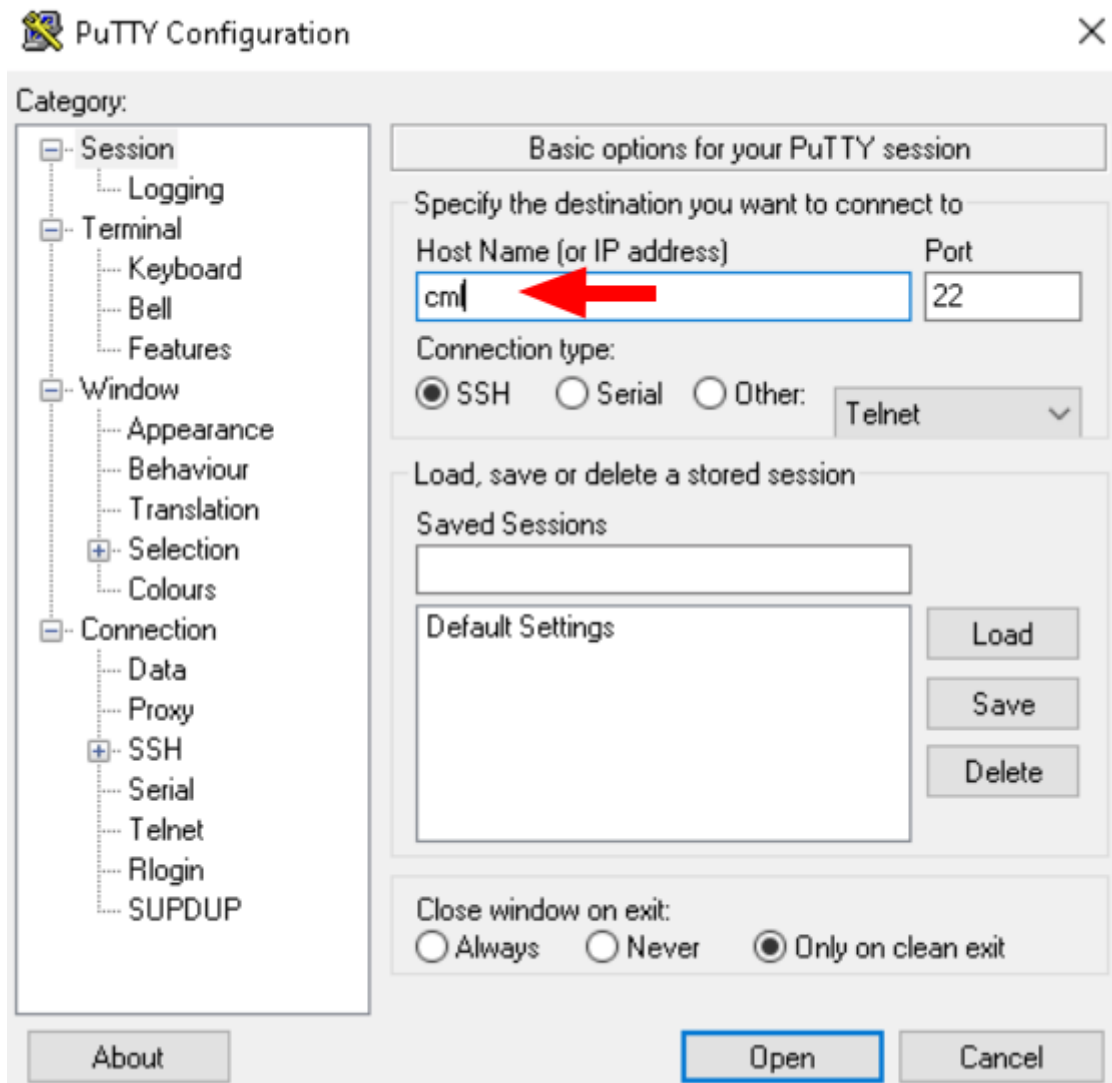
This concludes the breakout section.

Terminal Server

The built-in terminal server of CML allows secure and authenticated access to otherwise non-authenticated and plain text serial consoles of virtual network devices.

CML uses SSH on the standard SSH port TCP/22 to provide either a menu or, for advanced use cases, direct access to individual consoles of devices. The CML terminal server replaces the standard SSH server of the underlying OS. The OS SSH server is disabled by default. When enabled via Cockpit (as shown in one of the previous sections) the OS SSH server listens on TCP/1122.

Use the PuTTY client on Windows and point it to the CML controller using `cml` as the host name:



Leave SSH selected for the connection type. Click "Open" (you might get a warning about the key fingerprints of the host that needs to be confirmed).

```
cml-controller.cml.lab - PuTTY
login as: admin
admin@cml-controller's password:

****
CML2 Console Server
Copyright (c) 2019-2023 Cisco Systems, Inc. and/or its affiliates
****

tab completion works
list available nodes and node labels / IDs with "list"
it's also possible to do a "open /Lab at Wed 22:21 PM/iosv-0/0" command
or to directly connect via UUID "connect 697bb5b8-7181-49c5-8e85-540eacf9deac"
(e.g. lab name followed by node name and line number

consoles> █
```

Enter user name (`admin`) and password (`C1sco12345`) and you will be greeted by the terminal server prompt. From here, you can list labs and open consoles. Note that you can also press tab to complete names. Start with typing `help`.

```
cml-controller.cml.lab - PuTTY

Lab                                     Node      Lines
-----
test                                   alpine-0  0
consoles> open /test/alpine-0/0
Connecting to console for alpine-0
Connected to terminalserver.
Escape character is '^]'.

Welcome to Alpine Linux 3.16
Kernel 5.15.68-0-virt on an x86_64 (/dev/ttyS0)

inserthostname-here login: cisco
Password:
Welcome to Alpine!

The Alpine Wiki contains a large amount of how-to guides and general
information about administrating Alpine systems.
See <http://wiki.alpinelinux.org/>.

You can setup the system with the command: setup-alpine

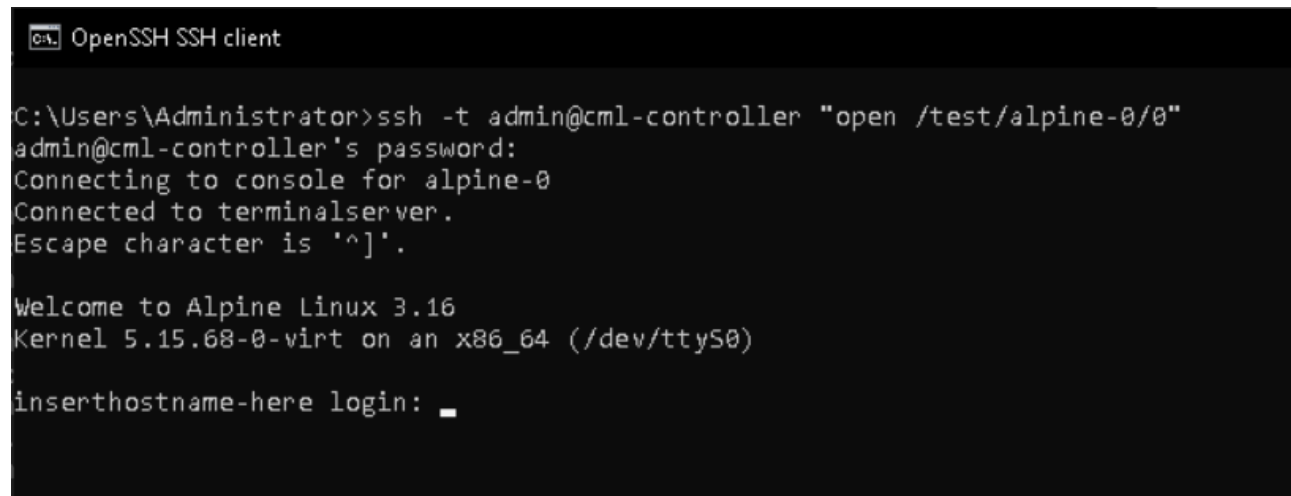
You may change this message by editing /etc/motd.

inserthostname-here:~$ █
```

press Tab key after typing "open"

Advanced usage

The terminal server also allows direct access to device consoles, omitting the menu. You can directly pass the "command" to the SSH client.

A screenshot of a terminal window titled "OpenSSH SSH client". The prompt is "C:\Users\Administrator>". The command entered is "ssh -t admin@cml-controller 'open /test/alpine-0/0'". The output shows the password prompt, connection status, and the Alpine Linux login prompt.

```
C:\Users\Administrator>ssh -t admin@cml-controller "open /test/alpine-0/0"
admin@cml-controller's password:
Connecting to console for alpine-0
Connected to terminalserver.
Escape character is '^]'.

Welcome to Alpine Linux 3.16
Kernel 5.15.68-0-virt on an x86_64 (/dev/ttyS0)

inserthostname-here login: _
```

The important piece here is to pass the "-t" flag to the SSH client. This requests a TTY for the session and is mandatory to get an interactive prompt. Note that the `open /test/alpine-0/0` is identical to the command that is executed in interactive mode above.

The same procedure is also possible with e.g. PuTTY but it's easier and conciser to demonstrate with the OpenSSH command line client.

This concludes the terminal server section.

Appendix

Additional tools and links

- Ansible plug-in to automate labs is available in Ansible galaxy or [here](#).
- Terraform provider available on [Hashicorp's registry](#) or [here](#)
- PyATS for automated device testing [here](#)
- CML Community repository with additional node definitions and utilities [here](#)

Other custom images

Windows 10

Microsoft used to provide Edge Developer VMs based on Windows 10. These have been, however, removed. There's still eval VMs for Windows 11 available but CML currently lacks TPM support... and, beware, these images are quite big.

Virtual WLC

As an alternative for the custom image, you can also install the virtual Wireless LAN Controller from [CCO here](#), choose the OVA ~620MB, "Cisco Wireless LAN Small Scale Virtual Controller Installation with 60 day evaluation license". The process is slightly different as we need to "pre-install" the operating system.

File Information	Release Date	Size
Cisco Wireless LAN Small Scale Virtual Controller upgrade. AIR-CTVM-K9-8-10-185-0.aes Advisories	24-Mar-2023	494.02 MB
Cisco Wireless LAN Small Scale Virtual Controller Installation with 60 day evaluation license. AIR-CTVM-K9_8_10_185_0.ova Advisories	24-Mar-2023	621.49 MB
Cisco Wireless LAN Large Scale Virtual Controller. AIR-CTVM_LARGE-K9_8_10_185_0.aes Advisories	24-Mar-2023	494.02 MB
Cisco Wireless LAN Large Scale Virtual Controller. AIR-CTVM_LARGE-K9_8_10_185_0.ova Advisories	24-Mar-2023	621.48 MB
Supplementary AP Bundle Images for Cisco CTVM_LARGE Series WLC Release 8.10. This bundle is mandatory to support AP9117 AP_BUNDLE_CTVM_LARGE_8_10_185_0.aes Advisories	24-Mar-2023	59.19 MB
Supplementary AP Bundle Images for Cisco CTVM_SMALL Series WLC Release 8.10. This bundle is mandatory to support AP9117 AP_BUNDLE_CTVM_SMALL_8_10_185_0.aes Advisories	24-Mar-2023	59.19 MB
Cisco Wireless LAN Small Scale Virtual Controller Installation with	24-Mar-2023	621.40 MB

vWLC Image Building

The vWLC building process is slightly different as the disk image finally deployed on CML must be created and the OS installed on it first. This is a common scenario where install media is provided which needs to be installed on a disk. For this reason, the steps here include the creation of a stand-alone VM using Qemu/KVM and, as a result of this process, obtain a disk file which then can be used inside of CML.

Note this might or might not work depending on the OS and what the installation process creates on the target disk. There's typically some "cleanup" required at the end of such an installation to allow the "cloning" of machines created from such a disk image.

It sometimes might be a good idea to install the desired OS inside of VMware Workstation or VirtualBox and, when done, convert the resulting disk into QCOW2 format.

These steps are for reference and can be done on the CML controller itself. Or on any Linux host that has the Qemu packages installed.

When using the `-curses` parameter the console of the VM will be in the terminal...

However, to kill/end the VM you need to get into another terminal and kill the process.

Without `-curses` and with a UI (like on the Mac or on a Linux with UI / Desktop) one can simply close the Qemu window.

```
$ mkdir build
$ cd build
$ tar xvf ../AIR*.ova
x AS_CTVM_SMALL_8_10_151_0.ovf
x AS_CTVM_SMALL_8_10_151_0.mf
x AS_CTVM_SMALL_8_10_151_0.vmdk
x AS_CTVM_SMALL_8_10_151_0.iso
x VmMgmtNetwork.xml
x VmSpNetwork.xml
$ qemu-img convert -pc -Oqcow2 AS_CTVM_SMALL_8_10_151_0.vmdk
AS_CTVM_SMALL_8_10_151_0.qcow2
(100.00/100%)
$ qemu-system-x86_64 -curses -accel kvm -m 2048 -cdrom
AS_CTVM_SMALL_8_10_151_0.iso -hda AS_CTVM_SMALL_8_10_151_0.qcow2
qemu-system-x86_64: warning: host doesn't support requested feature:
CPUID.80000001H:ECX.svm [bit 2]
$ ls -lh
total 4130984
-rw-r--r--@ 1 rschmied  staff   618M Mar  4 10:57
AS_CTVM_SMALL_8_10_151_0.iso
-rw-r--r--@ 1 rschmied  staff    366B Mar  4 10:57
AS_CTVM_SMALL_8_10_151_0.mf
```

```

-rw-r--r--@ 1 rschmied  staff    6.4K Mar  4 10:57
AS_CTVM_SMALL_8_10_151_0.ovf
-rw-r--r--  1 rschmied  staff    1.4G Jul 14 17:32
AS_CTVM_SMALL_8_10_151_0.qcow2
-rw-r--r--@ 1 rschmied  staff     72K Mar  4 10:57
AS_CTVM_SMALL_8_10_151_0.vmdk
-rw-r--r--@ 1 rschmied  staff    1.3K Mar  4 10:57 VmMgmtNetwork.xml
-rw-r--r--@ 1 rschmied  staff    1.0K Mar  4 10:57 VmSpNetwork.xml
$ qemu-img convert -pc -Oqcow2 AS_CTVM_SMALL_8_10_151_0.qcow2 vwlc-
disk.qcow2
      (100.00/100%)
$ rm AS* Vm*.xml
$ ls -lh
total 2234368
-rw-r--r--  1 rschmied  staff    1.1G Jul 14 17:34 vwlc-disk.qcow2
$

```

This can be used to test the image. It will create a linked clone of the disk named `test-disk.qcow2` and start a VM... You should see output on the screen... You can also `telnet 127.0.0.1 4444` to get access to the serial console. Same caveat regarding closing the VM applies (see above).

```

$ qemu-img create -b vwlc-disk.qcow2 -fqcow2 test-disk.qcow2
$ qemu-system-x86_64 -curses -accel hvf -m 2048 -hda test-disk.qcow2
-serial telnet:127.0.0.1:4444,server,nowait -device e1000

```

Running your own instance

Hardware

You need a computer to run CML on. A somewhat recent laptop should be sufficient to get you up-and-running. As laptop are somewhat limited in available resources, but they should be sufficient to learn the basics and to get a better understanding how to install and run CML on larger computers.

- **Memory:** At least 16GB of memory, 8 or 10GB should be dedicated to the CML Virtual Machine
- **CPU:** the more, the merrier... 4 vCPUs / threads should be assigned to the CML Virtual Machine
- **Disk Space:** At least 8GB of free disk space required, more is better especially if you want to install additional images like Windows 10... For Windows 10, you should set at least 16GB of free space aside.

Recommendation: Use a USB 3.0 portable hard drive to store the images and install the software on.

License

To run the software, you also need a license. Personal or Personal plus licenses can be bought via the [Cisco Learning Network Store](#).

Software

VMware Workstation

CML comes as an OVA to be deployed on VMware products. To run CML, you need to have a hypervisor installed. On Windows and Linux, VMware Workstation can be used. Unfortunately, running CML on Apple hardware is only possible with Intel architecture, Apple Silicon models can't run CML! Alternatively, you can run on VMware ESXi 6.7 or later as well if you have access to such infrastructure.

You should also be OK to use a trial license for Workstation or use VMware Player, if available on your platform.

Cisco Modeling Labs Software

Software can be downloaded from CCO. You need two images, the OVA and the ISO. Bare metal installs (as an alternative to install as a virtual machine) require the ISO instead of the OVA.

<https://software.cisco.com/download/home/286193282/type/286326381>

Download the OVA and the reference platform ISO file (here, the 2.5.1 version is shown):

Search...

Expand AllCollapse All

Suggested Release

2.5.1

Latest Release

2.5.1

2.4.1

2.3.1

2.2.3

All Release

2.5

2.4

2.3

2.2

Modeling Labs

Release 2.5.1

My Notifications

Related Links and Documentation

No related links or documentation

File Information	Release Date	Size
Cisco Modeling Labs 2.5.1 server. This image is for bare metal deployment. For VMware Installation use the (.ova) VMWare Installation Image found of Software Center. cml2_2.5.1-10_amd64-5.iso.zip Advisories	14-Apr-2023	1397.91 MB
Cisco Modeling Labs 2.5.1 server (upgrade). This file is used to upgrade CML 2.3.x and CML 2.4.x installations to CML 2.5.1. Please follow the instructions in the CML Installation Guide to upgrade a CML server. cml2_2.5.1-10_amd64-5.pkg.zip Advisories	14-Apr-2023	155.30 MB
Cisco Modeling Labs reference platform ISO file (March 2023). This file is a required file for users who are *not* migrating from an existing installation. refplat-20230117-fcs.iso.zip Advisories	14-Apr-2023	10193.28 MB
Cisco Modeling Labs 2.5.1 server. This image is for deployment on VMware. Supported Hypervisors can be found in the CML Installation Guide. For Bare Metal Installation use the bare metal (.iso) installation file found in software center. cml2_2.5.1-10_amd64-5.ova Advisories	12-Apr-2023	836.35 MB