Using and Configuring Cisco Modeling Labs 2.0

First Published: 2020-04-14

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

CHAPTER 1

Introduction to Cisco Modeling Labs

Overview of CML 2.0

Cisco Modeling Labs 2.0 is a major update of the entire Cisco Modeling Labs (CML) network simulation platform. While the platform still uses KVM as the hypervisor to run the same network OS virtual machine (VM) images, we have completely rewritten the rest of the platform. For example, we replaced the desktop GUI application with a new HTML5 browser-based user interface (UI). The software that orchestrates and runs the simulation is brand new and has a much smaller memory footprint. We greatly simplified the installation and initial simulation creation to improve the user experience. The virtual machines in the network simulations are connected via a custom-designed fabric. These changes provide for a more secure, easier-to-use network simulation platform and enable new core concepts in the product.

Starting with CML 2.0, you can think of each of your network topologies as a *lab*. You create and modify your labs on the CML server. With some limitations, you can modify the topology while the lab simulation is running. For example, you can change the connections between nodes, and you can add new nodes and connect them to the topology without stopping the simulation. Labs are also persistent by default now, unlike in the 1.x versions of the product. That is, when you stop a simulation, the disk images for the VMs in the lab are not discarded. This persistence preserves the state of each node, including crypto keys, license keys, and newly-installed packages.

CML 2.0 is built on top of REST-based web service APIs designed with both security and automation in mind. You can use these APIs to create labs and drive the entire simulation lifecycle programmatically. The new release was designed "API first" to ensure that fine-grained operations are exposed via the APIs in a consistent way. The product uses these APIs in its own user-facing interfaces:

- the HTML5 UI
- companion utilities, such as the Breakout Tool
- the Python client library

CML enables you to create and run virtual networks. You can use these labs for personal study for certification, for teaching networking classes, and for testing out new protocols or configuration changes. With the changes in the 2.0 release, CML also becomes part of a larger NetDevOps ecosystem, enabling you to test and validate network changes in an automated workflow. CML 2.0 is a complete rewrite of the product and introduces fundamental changes. If you use CML 1.x or Cisco VIRL Personal Edition 1.x, then we recommend that you read the entire CML 2.0 release notes before you get started.

Using CML and the HTML5 UI

Cisco Modeling Labs is a network simulation platform. You can use CML to create lab networks and run simulations of those labs. Once you have a running lab simulation, you can interact with the VMs in the lab just as you would with devices in a real network. This guide documents the CML HTML5 UI and the use of the Breakout Tool, on page 25, which gives you local access to consoles of the nodes in the remote lab. For information on administrative tasks, such as installation and licensing, see *Administering Cisco Modeling Labs 2.0*.

To open the CML UI, use a supported web browser to visit the URL for the CML server. The URL is shown in the message above the login prompt in the VMware console for the CML server's virtual machine: "Access the CML UI from https://nnn.nnn.nnn.nnn/". The CML UI requires you to log in using an application account. During installation, an *initial user* was created that you can use to log into the UI. If you are using CML-Enterprise, ask the application administrator for the credentials to use with CML.

The primary pages for working with labs in the CML UI are the **Lab Manager** and the **Workbench**. The **Lab Manager** provides access to all of the labs that you have created and the ability to navigate to the other pages of the UI. Clicking on a lab opens it in the **Workbench**. You edit a lab and interact with the lab's simulation on the **Workbench** page. There is a menu bar at the top of each page that provides easy access to navigation controls, your user menu, and any additional menus or actions that are relevant to the current page. You can return to the **Lab Manager** page at any time by clicking on **Lab Manager** in the menu bar or by clicking on the Cisco logo at the top-left corner of the page.

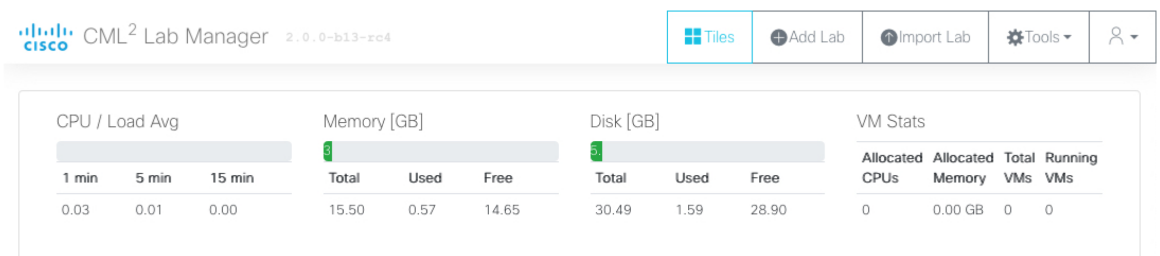
CHAPTER 2

Lab Manager

The Lab Manager Page

Lab Manager is the landing page that you see when you first log into the UI. At the top of the **Lab Manager** page, there is a summary of the CML server's system status.

Figure 1: System Status



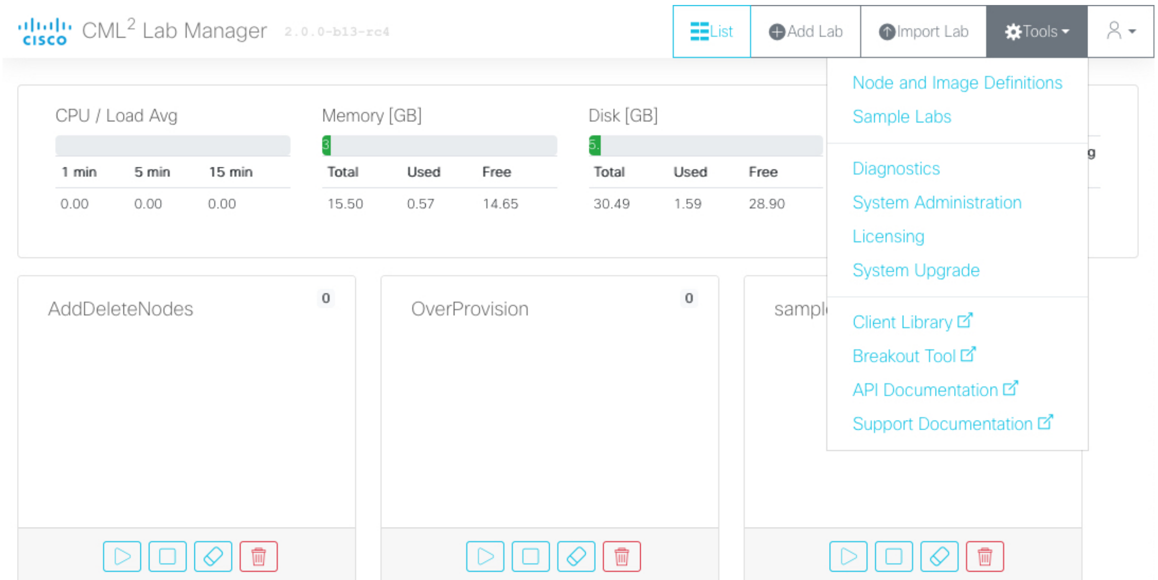
Below the system status elements, the **Lab Manager** shows a list of the labs that you have created. By default, the list of labs is displayed as a set of tiles. If you have not yet created a lab, no lab tiles will be displayed. Clicking on a lab tile opens that lab in the Workbench.

There is a menu bar at the top of the page with the following items:

Menu Item	Button Type	Tasks
List (default)	toggle	Changes the layout of the labs. You can toggle between **Tiles** and **List** view by pressing the button. See Changing the Lab List View and Multi-Selecting Labs, on page 6.
Add Lab	action	Creates a new lab. See Creating a Lab, on page 5.
Import Lab	action	Opens the **Import Lab** page. See Importing Labs, on page 5.
Tools	menu	Opens the tools menu.

The **Tools** menu item opens the **Tools** menu.

Figure 2: Tools Menu



Lab Manager and **Workbench** are the primary pages in the UI for working with labs. Other pages of the user interface are accessible via the **Tools** menu:

Menu Item / Page Name	Description
Node and Image Definitions	The **Node and Image Definitions** page shows the node definitions and images available on this CML server. You can create new node definitions and add VM images on this page. See Preparing Custom Images, on page 33
Sample Labs	The **Sample Labs** page lists preconfigured labs that are ready to use. Load a sample lab into the **Lab Manager** to get acquainted with the new CML interface.
Diagnostics*	The **Debug Diagnostics** page provides logging information that is useful for troubleshooting.
System Administration*	The **System Administration** page provides system information and links for accessing the CML system settings. The administrator adds new CML application users on this page.
Licensing*	Use the **Smart Software Licensing** page to apply a license or to view the license authorization status.
System Upgrade*	See *Administering Cisco Modeling Labs 2.0* for more details on system upgrades.
Client Library	Opens the documentation page for the Python Client Library for CML.
Breakout Tool	Opens the documentation page for the Breakout Tool. Download the Breakout Tool from this page. See Breakout Tool, on page 25.
API Documentation	Opens the documentation for CML's web service APIs.

Menu Item / Page Name	Description
Support Documentation	Opens a page with information about getting support for CML.
* = These items are only available to users with application administrator privileges.	

Creating a Lab

Before you begin

Log into the CML web UI using an application user account. For example, you may use the *initial user* account created during the installation process. After logging in, you will see the **Lab Manager** page of the UI.

Step 1 On the **Lab Manager**'s menu bar, click **Add Lab**.

A new lab is created, and the tile for the lab will appear in the **Lab Manager** with a *default* name, such as `Lab at Tue 14:30 PM`.

Step 2 **Optional:** Hover over the lab tile and click the lab name to start editing.

Step 3 **Optional:** Enter a new lab name and press **Enter** to rename the lab.

You may edit the lab name at any time from the **Lab Manager** tile or in the **Workbench**'s menu bar.

Importing Labs

The **Import Lab** page supports the import of both CML 2.0 .yaml topology files and 1.x .virl topology files.

When importing a .virl topology from a 1.x system, be aware of the following limitations:

- Images being used in the imported topology may not be available on the CML 2.0 system.
- The network configuration for the imported lab may not match the CML 2.0 system and may require further editing.
- Some images are no longer supported in CML 2.0. For example, the Ostinato node type has been replaced by Cisco TRex.

Tip CML will attempt to map unsupported node types from a 1.x .virl file to the closest matching node definition. For example, Ostinato drone nodes will become TRex nodes in the imported topology. Note that the import only converts the node to the new node definition. It cannot convert the node's associated configuration. To ensure that the topology functions as expected, you must check the topology and especially the configuration associated with the nodes. Manually update the configuration of the converted nodes to implement the desired behavior.

Step 1 From **Lab Manager**, select **Tools** > **Import Lab**.

Step 2 Click **Browse** and use the file browser to select the topology file to import from your local system.

Step 3 Click **Upload Topology**.

The topology is imported, and the page displays a message, such as `Uploaded as lab abc123`.

If you are importing a 1.x .virl file, you may also see warnings, such as

```
Warnings
Node /lxc-ostinato-drone-1: lxc-ostinato-drone node mapped to trex, manual adaption
required!
Node /lxc-1: lxc node mapped to alpine, manual adaption required!
```

Pay attention to any warnings displayed during import. Warnings indicate that the topology contains features or node types that have changed since version 1.6. You should check the imported topology before running it and make necessary changes to the node configurations, especially for any nodes mentioned in the warnings.

Step 4 Click **Go to Lab...** to open the newly imported lab in the **Workbench**, or click **Lab Manager** to return to that page and see the new lab in the list of labs.

Organizing Labs

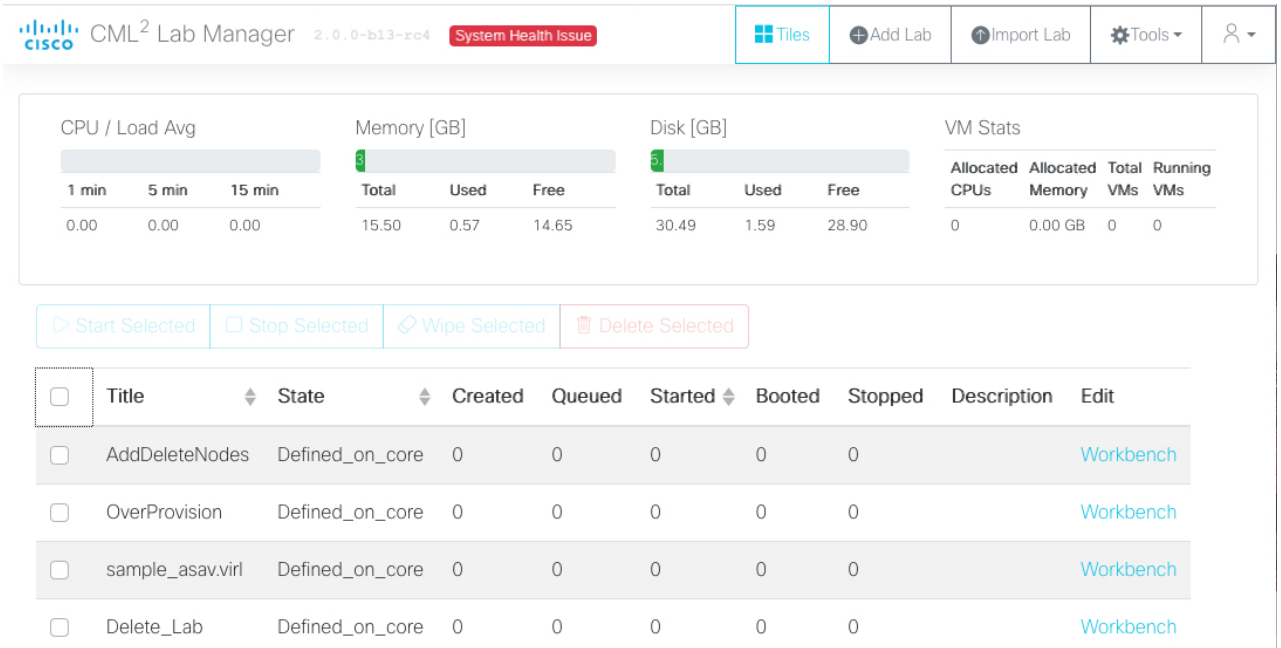
Changing the Lab List View and Multi-Selecting Labs

Lab Manager uses a tiles-based view to show the list of labs. You may change the **Lab Manager**'s layout from tiles view to list view. The list view permits the selection of multiple labs, but the tiles view does not. To apply an action to multiple labs at once, you must first switch to list view.

Step 1 Click the **List** button on the menu bar to change the **Lab Manager**'s layout to list view.

After the button is pressed, **Lab Manager** shows the labs in a list. The button's label changes to **Tiles**.

Figure 3: List view for Lab Manager

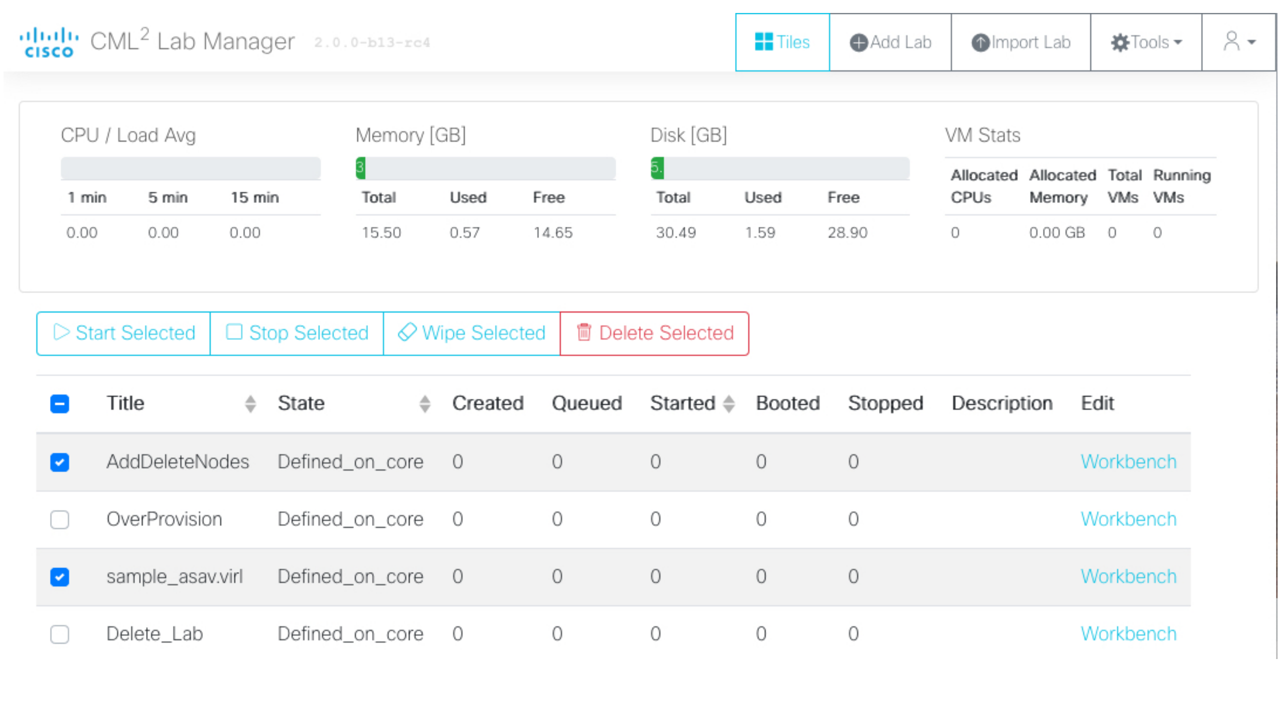


Step 2 Select one or more labs by checking the check box in that lab's row.

With multiple labs selected, you can perform an action (start, stop, wipe, or delete) on all of the selected labs.

Example:

Figure 4: Selecting multiple labs in the Lab Manager

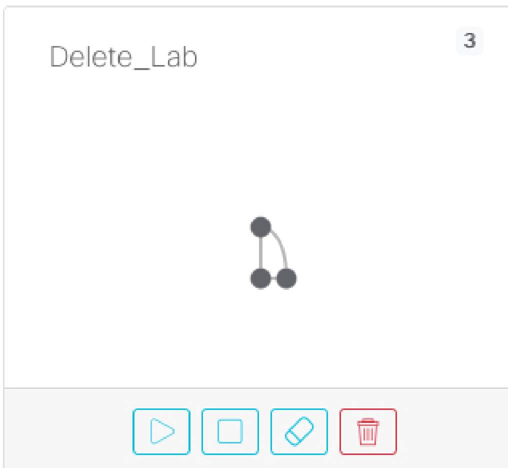


Deleting Labs

Labs in CML are persistent. After a lab has been stopped, CML will maintain the state for all nodes in the lab. Any setting that you modify during the simulation is automatically saved by the system. The next time you start the lab, all nodes will be using the same running configurations from the last time the lab was stopped.

To delete a lab, you must first stop the lab simulation and remove (wipe) its persistent state.

Figure 5: Start Lab / Stop Lab / Wipe Lab / Delete Lab



Step 1 In the **Lab Manager** page, locate the lab that you want to delete. If the lab is running, the **Wipe Lab** button will be disabled and grayed out. Click the **Stop Lab** button first to stop the running lab.

Step 2 With the lab stopped, click the **Wipe Lab** button.
A confirmation dialog is shown.

Step 3 Click **OK** in the dialog to confirm that you want to wipe the lab.

By wiping the lab, all data associated with the lab will be permanently deleted.

Step 4 Click the **Delete Lab** button to remove the lab.
A confirmation dialog is shown.

Step 5 Click **OK** in the dialog to confirm that you want to permanently delete the lab.
The lab is deleted, and it is no longer shown in the list on the **Lab Manager** page.

CHAPTER 3

Workbench

The **Workbench** is used to create and modify a lab's topology and to interact with the lab simulation. To open a lab in the **Workbench**, find the lab in the **Lab Manager** page and, depending on the **Lab Manager**'s view, click on the tile (tiles view) or the *workbench* link (list view) associated with the lab.

Adding Nodes to a Lab

Before you begin

Open the lab that you want to edit in the **Workbench** by clicking on its tile on the **Lab Manager** page.

Step 1 Click the **Add Nodes** drawer in the **Workbench** to open it.

In a new lab or a lab with no nodes, the **Add Nodes** drawer will already be open by default.

Step 2 Drag-and-drop a node type from the **Add Nodes** drawer onto the lab to add a node of that type to the lab's network topology.

Step 3 Repeat the previous step for each node that you want to add to the lab.

Starting, Stopping, and Wiping Nodes

When starting a lab simulation, CML starts a virtual machine (VM) for each node in the lab. Similarly, stopping a lab or wiping a lab affects all of the nodes in the lab. If you want to start, stop, or wipe individual nodes, you can do that in the **Workbench**. If you are running a simulation, you can start, stop, or wipe individual nodes in that simulation without stopping the entire lab or interrupting the other nodes in the simulation.

Step 1 In the **Workbench**, select the node.

Step 2 Click the **Simulate** tab in the bottom pane of the **Workbench**.

Step 3 Click the **Start**, **Stop**, or **Wipe Node** button in the **Simulate** pane.

Caution Pay attention to the button labels. If you have not selected a single node, then the buttons in the **Simulate** pane will indicate **Start Lab**, **Stop Lab**, or **Wipe Lab**. Clicking on these buttons will apply the action to *all* nodes in the lab.

Start

Starts a VM associated with the node.

Starting a node allocates a VM for the node if one does not already exist. If the node has been started before, it reuses the already-allocated resources, such as disk images and MAC addresses.

Stop

Stops the VM associated with the node.

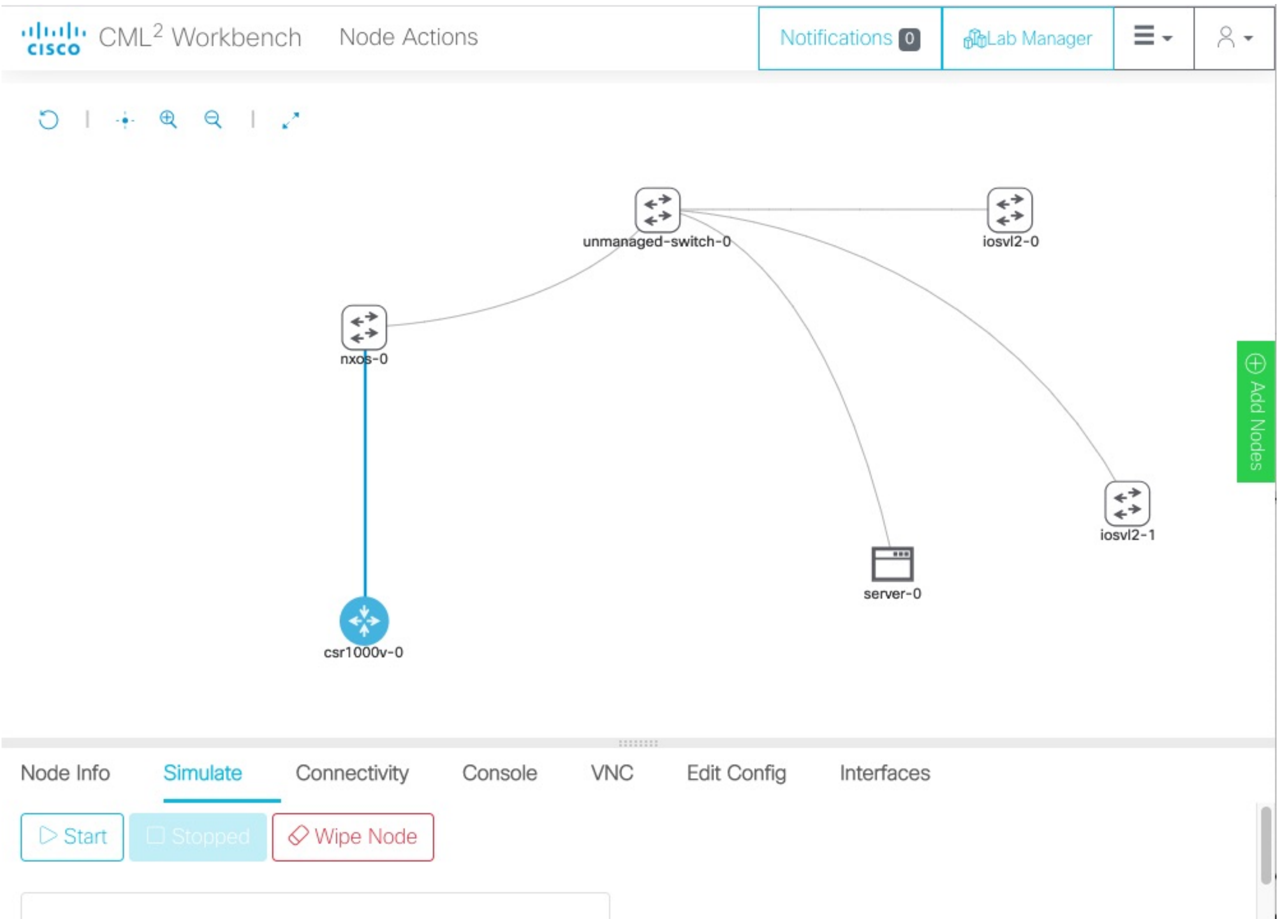
Stopping a node releases its memory and CPU resources. A stopped node is still present on the CML server and retains its disks and other properties for use when it is started again.

Wipe Node

Deletes all state associated with the node.

Wiping a node removes all state and properties associated with the node from the CML server. The next time you start the node, CML will re-create it and assign new MAC addresses, disks, and other resources. Since a wiped node has no pre-existing state, the node will load its bootstrap configuration the first time it is restarted. The node's bootstrap configuration can be found in the **Edit Config** pane of the **Workbench**.

Figure 6: The Workbench's Simulate pane with Start, Stop Lab, and Wipe actions



Depending on the node's status, one or more of the buttons in the node's **Simulate** pane may be disabled. The **Start** and **Wipe Node** buttons are only enabled when the node is already stopped. The **Stop** button is only enabled if the node has been started.

Related Topics

Starting a Simulation, on page 18
Stopping a Simulation, on page 20

Deleting Nodes

Before a node can be deleted, the node's associated state must first be removed. If you have ever run a simulation with the node, then the node's state persists even when the simulation stops. CML requires you to wipe a node before deleting it to ensure that you do not accidentally delete a node that still has some associated state that you want to preserve.

Step 1 In the lab **Workbench**, select the node to be deleted.

Step 2 Click the **Node Info** tab in the bottom pane of the **Workbench**.

Step 3 Click the **Delete Node** button in the **Node Info** pane.
A popup confirmation dialog is shown.

A node can only be deleted if it is currently stopped and has no associated state on the CML server. If the **Delete Node** button is disabled, first stop and wipe the node. See Starting, Stopping, and Wiping Nodes, on page 10.

Step 4 Click **OK** in the dialog to confirm that you want to delete this node and its connections from the lab.

Creating Links

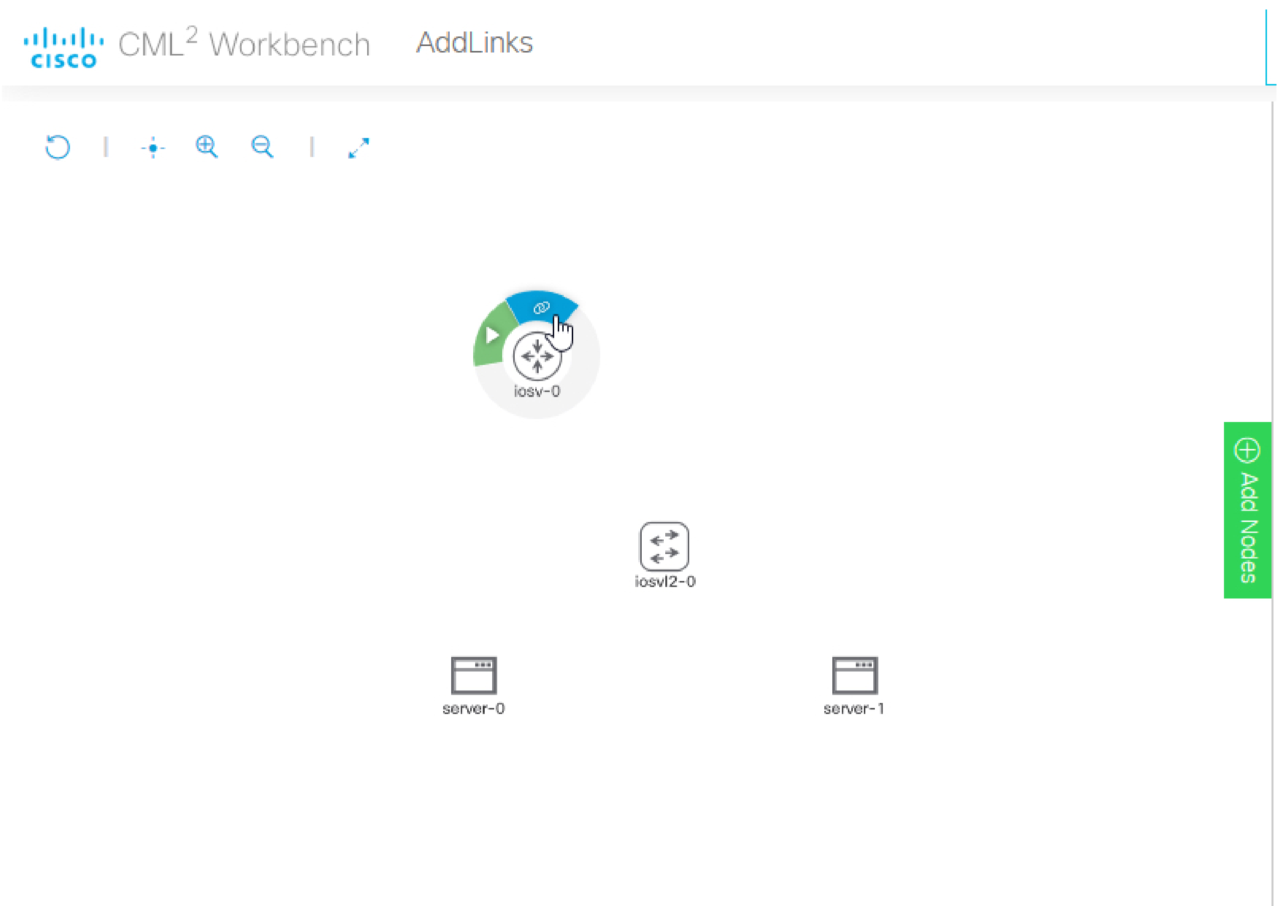
Links function like a physical wire that connects two interfaces. Link the nodes in your lab to create the desired network topology.

Step 1 In the **Workbench**, hover over the desired *source* node for the new connection to activate the **action ring** for the node.

Step 2 Press and hold the mouse button on the blue **link** button.

Example:

Figure 7: G



Note: if the mouse cursor changes to a *cancel* icon when you hover over the link button on the **action ring**, the **Workbench** is indicating that that the source node has no more available interfaces. You will not be able to create any additional connections to or from this node until an existing interface is freed.

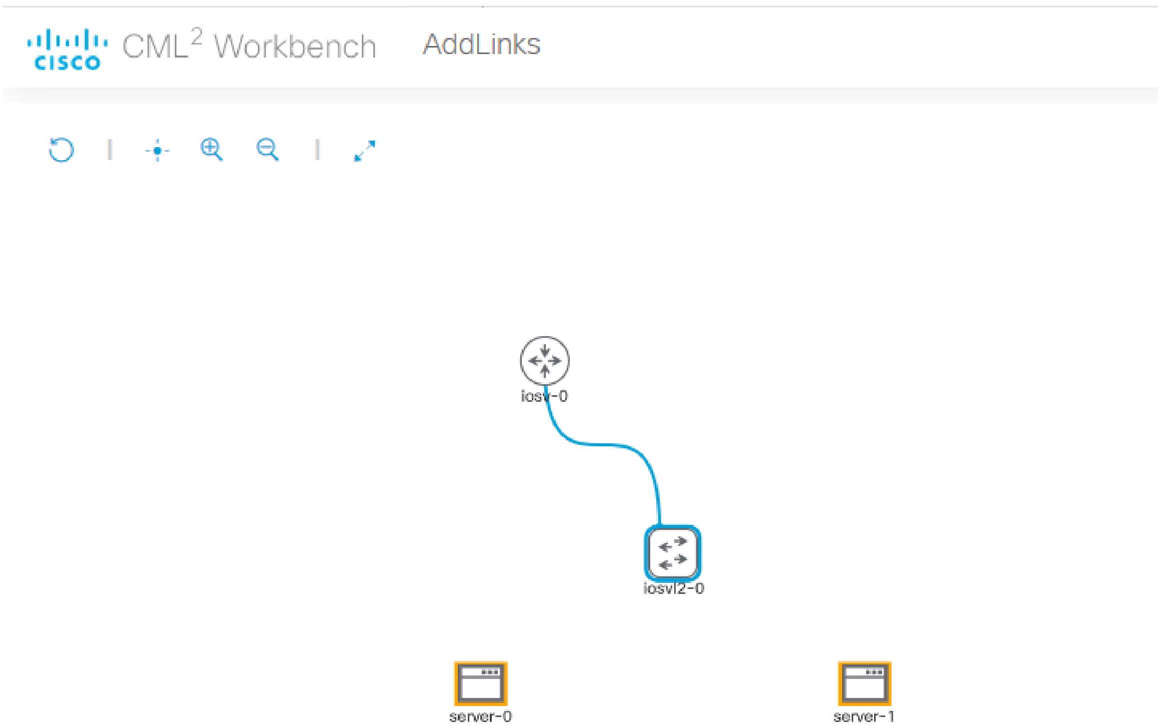
Step 3 While holding the mouse button, drag away from the source node to start creating a connection.

Step 4 Hover the mouse over the desired *destination* node for the connection and then release the mouse button.

The currently selected node will be highlighted in blue.

Example:

Figure 8: Selecting the destination node for a link in the Workbench



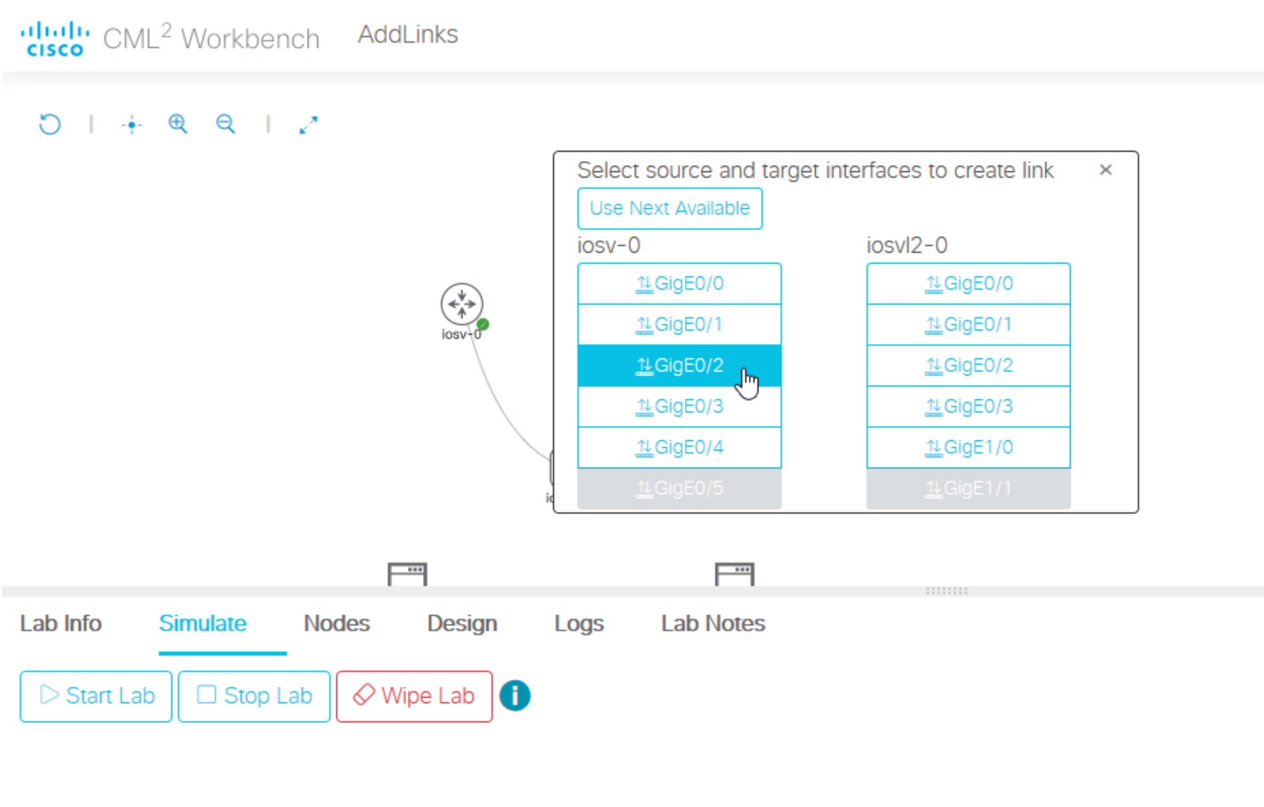
The **Workbench** pops up an interface selection dialog.

Note: the currently selected node should be highlighted in blue. If the selected node is highlighted in red, and the mouse cursor changes to a *cancel* icon, the **Workbench** is indicating that the destination node has no more available interfaces. You will not be able to create any additional connections to or from this node until an existing interface is freed.

Step 5 Select the desired interface on the *Source* and *Destination* nodes in the dialog. Alternatively, you can just click the **Use Next Available** (interface) button, and the connection will be created with the first unused interface on each node.

Example:

Figure 9: The unused and provisioned interfaces for nodes in the interface selection dialog



Related Topics

Rules for Creating Links and Interface Overprovisioning, on page 14
Adding Interfaces to a Node, on page 15

Rules for Creating Links and Interface Overprovisioning

While you can create or redirect links even when the nodes involved in the link are running as part of a simulation, it is only possible to create a link if the nodes at either end of the link each have an unused interface available. Once a node has been started, the number of interfaces allocated to that node cannot be changed until the node is stopped *and* wiped. This restriction means that link creation and interface creation behave differently once you have started a node.

If you add a node to the canvas and have not started it yet, or if you have wiped a node and not started since wiping it, the **Workbench** automatically adds additional interfaces to a node to accommodate new links. You may also add extra interfaces directly to a node. Each node definition has a maximum possible number of interfaces. If the node is not at the maximum number of allocated interfaces for its node definition, you can select it as the source or destination of a link. If all of the allocated interfaces are used by existing links, the **Workbench** permits you to select an unallocated interface for the connection, and it allocates that interface to the node when you create the connection. If the node already has its maximum number of interfaces allocated, and all of the interfaces are in use, then it is not possible to create a new link with that node.

If a node is currently running as part of a simulation, or if the node is stopped but has associated simulation state that has not been wiped, then the number of interfaces allocated to that node cannot be changed. If at least one of the interfaces allocated to the node is unused, then you can select it as the source or destination of a link. If all of the interfaces allocated to the node are used by existing links, then it is not possible to create a new link with that node even if the node has fewer allocated interfaces than its maximum number of allowed interfaces. Before you can add a link or interface to such a node, you must stop and wipe the node.

Tip To avoid having to stop a node in your simulation or to recreate its state after wiping it, you may want to **overprovision** interfaces on the nodes in your labs. Overprovisioning interfaces means allocating more interfaces on a node than it needs for the existing links in the topology. If you overprovision the interfaces on the nodes in your lab, each node has at least one unused interface when you first start the simulation. These extra interfaces enable you to add additional nodes and links to your lab later without having to stop or wipe any of the existing nodes in the lab.

Because interfaces consume system resources, CML only allocates additional interfaces to a node when the interfaces are needed for a new link. Each node definition has a default initial number of interfaces. When you first drag-and-drop a node into the lab, the node will have the default number of interfaces allocated to it. In smaller topologies, the default number of interfaces may create some modest overprovisioning, but if you want to overprovision interfaces in your lab, you should check all nodes before starting the lab to ensure that each one has enough unused interfaces allocated to each node.

Tip If you cannot create a connection to a node, select the node and then click the **Interfaces** tab in the bottom pane of the **Workbench**. If the button in the top-left of the **Interfaces** pane indicates **Max Interfaces Reached**, then the node has reached the maximum number of interfaces possible for that node type. If the button is disabled but indicates **Add # Interfaces**, then additional interfaces can be added to the node, but the node must be stopped and wiped first.

Related Topics

Starting, Stopping, and Wiping Nodes, on page 10
Adding Interfaces to a Node, on page 15
Creating Links, on page 12
Overprovisioning Interfaces with Link Creation, on page 16

Adding Interfaces to a Node

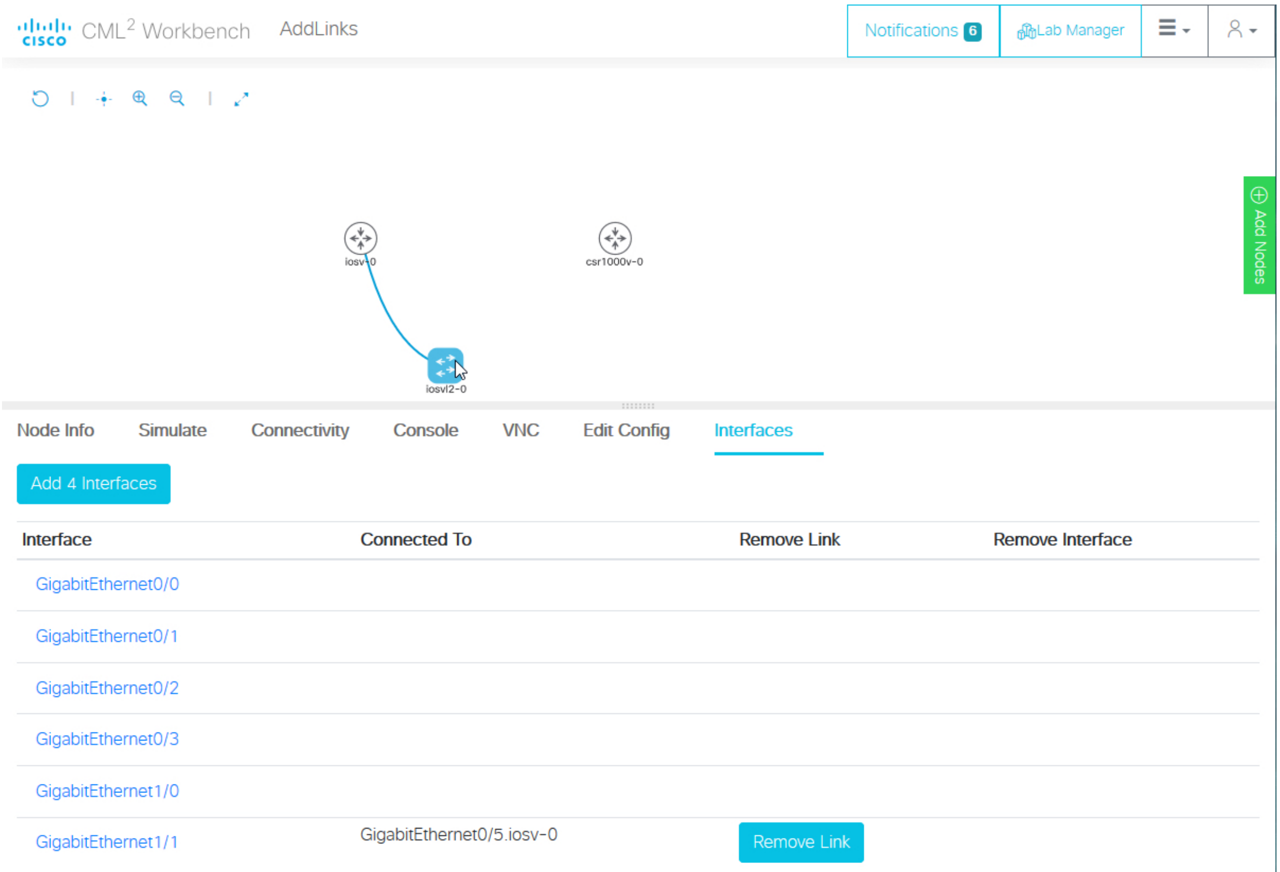
CML automatically allocates interfaces to a node to accommodate new links. It is not necessary to manually add interfaces unless you want to overprovision interfaces for the node. See also Rules for Creating Links and Interface Overprovisioning, on page 14.

Step 1 Select a node in the **Workbench**.

Step 2 Click the **Interfaces** tab in the bottom panel of the **Workbench**.

Example:

Figure 10: The Interfaces panel for a node in the Workbench.



Step 3 Click the **Add # Interfaces** button to add additional interfaces.

The button label shows how many interfaces will be added. By default, interfaces are added in groups of four, and the button shows **Add 4 Interfaces**. Each node definition has a maximum number of interfaces, and if the node can support fewer than four additional interfaces, the button label will change to indicate how many more interfaces can be added. If the node already has the maximum number of interfaces supported by its node type, then the button will be disabled and indicate **Max Interfaces Reached**.

Overprovisioning Interfaces with Link Creation

When Creating Links, the **Workbench** displays the interface selection dialog. Instead of clicking the **Use Next Available** button, you can select specific source and destination interfaces. The **Workbench** will allocate interfaces to the node to accommodate the new link, if necessary. If you always connect links to a node by using the next interface in sequence, once the default number of interfaces for the node are used, all of the interfaces allocated to the node will be used by a link. That is, when the node is started, it will have no interface overprovisioning, and no interfaces will be available for adding new links to that node at simulation time.

Instead of Adding Interfaces to a Node, you can also overprovision interfaces on a node while creating a link. Simply skip one or more interfaces in the interface list when you create a link.

Step 1 Follow the instructions for Creating Links until the interface selection dialog is shown.

Step 2 Select an interface from the interface list, skipping one or more unused interfaces.

Example:

For example, if you want to ensure that each node has five unused interfaces, you can start all of your interface selections at the sixth interface in the list, as shown. In this example, connect `GigE0/5` to `GigE1/1`.

Figure 11: Interface selection dialog for a new link in the Workbench

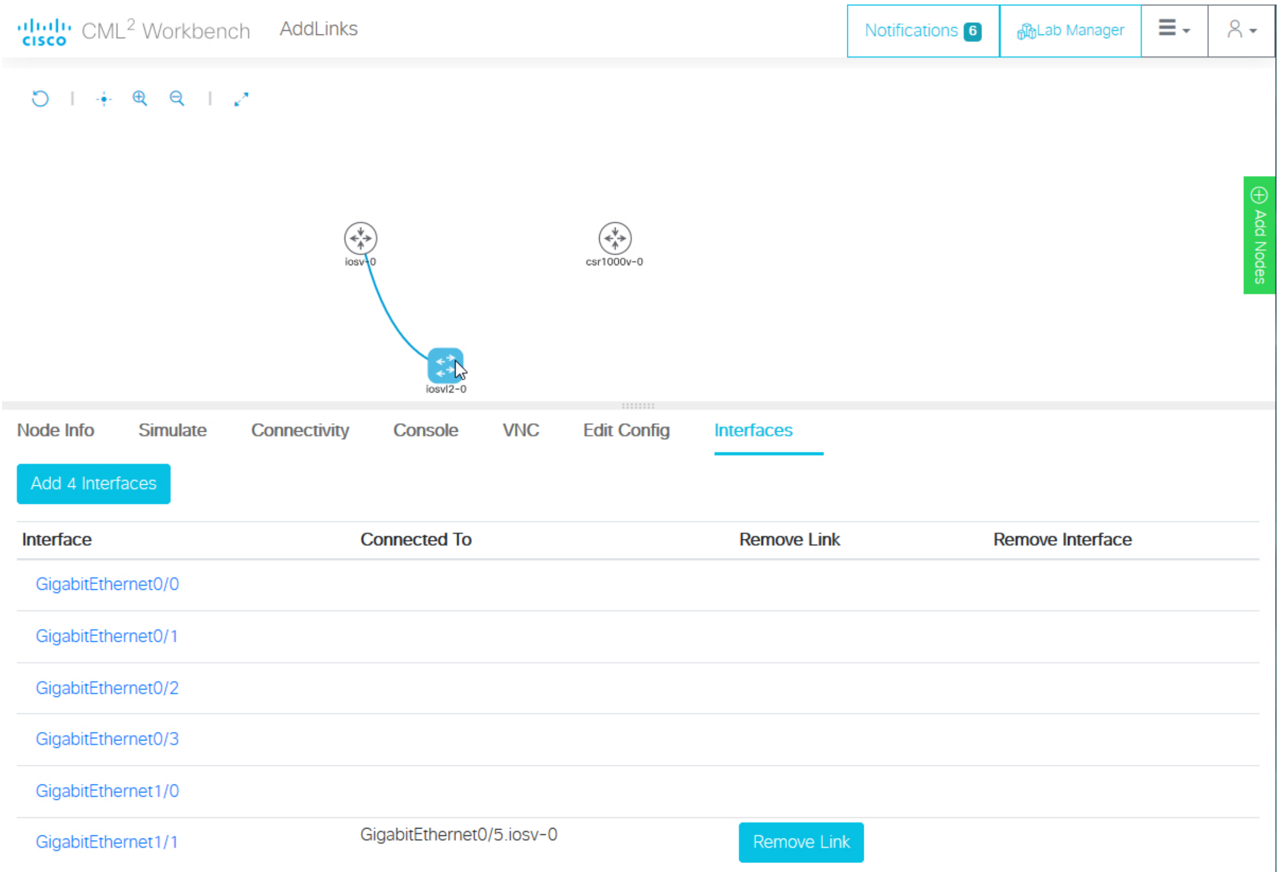
The link is created, and all interfaces up to the selected interface are allocated on the node.

Step 3 You can verify the number of allocated but unused interfaces by selecting a node and clicking on the **Interfaces** tab in the bottom pane of the **Workbench**. The **Interfaces** pane shows allocated interfaces. The interfaces with no listed connections are available for use.

Example:

In this example, interfaces `GigE0/0 - GigE1/0` are allocated but unused on iosvl2-0. These overprovisioned interfaces will be available for use when the simulation is running.

Figure 12: The Interfaces pane for a node in the Workbench



\

Tip If you select the last interface for a node in the interface selection dialog, the maximum number of interfaces for that node's definition will be allocated to the node.

Related Topics

Creating Links, on page 12
Rules for Creating Links and Interface Overprovisioning, on page 14

Starting a Simulation

Before you begin

Open the lab that you want to start in the **Workbench** by clicking on its tile on the **Lab Manager** page.

Step 1 First, ensure that no nodes are selected by clicking on an empty area of the **Workbench**.

Step 2 **Optional:** Select the **Design** tab at the bottom of the **Workbench** page.

Step 3 **Optional:** Click the **Build Initial Bootstrap Configurations** button in the **Design** panel.

You can also build configurations on an individual node by selecting the desired node and then clicking the **Build Initial Bootstrap Configurations** button.

CML generates a basic startup configuration for all nodes in the topology.

Step 4 Select the **Simulate** tab at bottom of the **Workbench** window.

Step 5 Click the **Start** button.
CML will start the simulation for the lab. Note that the **Workbench** will provide visual feedback, indicating each node's status. A node is finished booting and ready for use when the **Workbench** displays a green dot next to it.

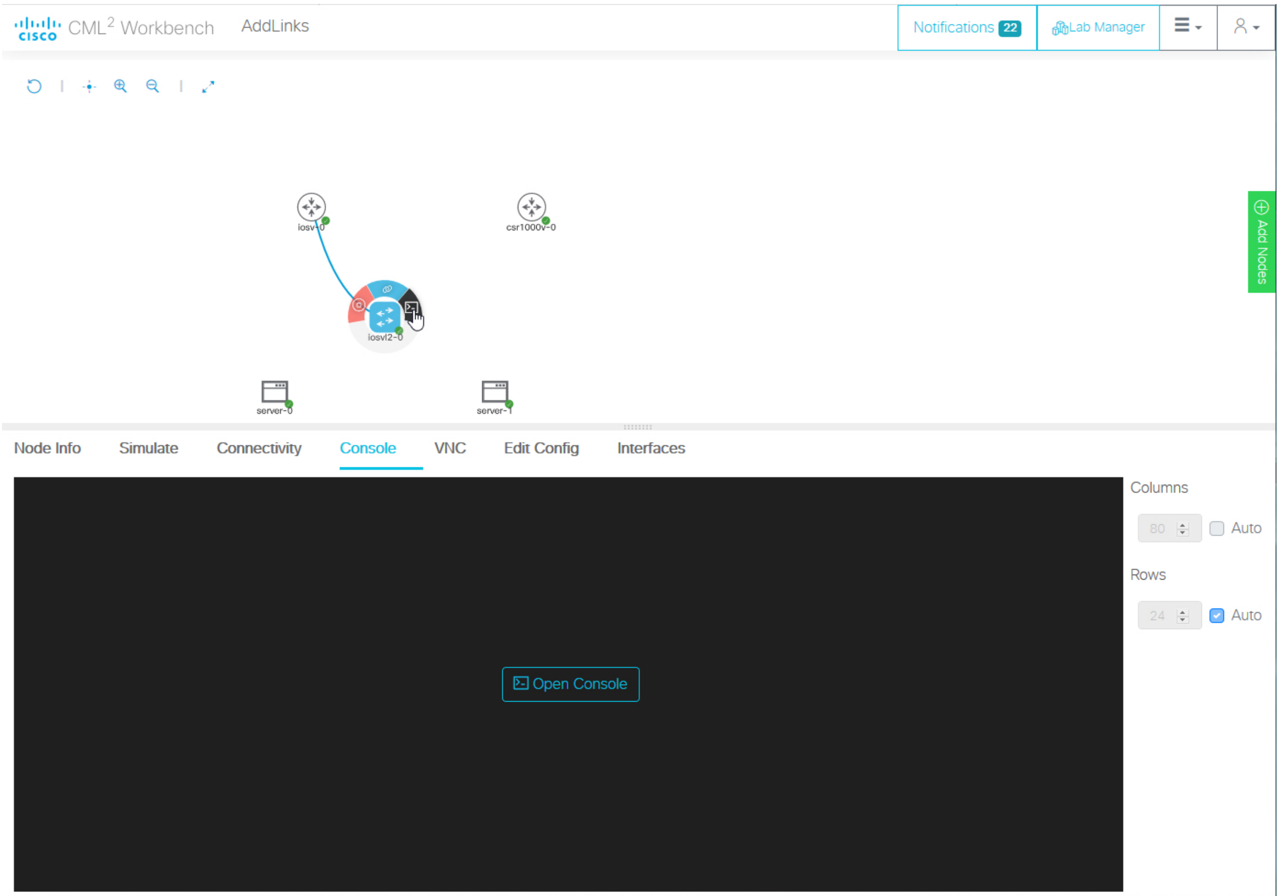
Connecting to a Node's Console

Once a simulation is started, you may connect to the console or, for applicable nodes, the VNC server for a node.

Step 1 Hover over the desired node to activate the node's **action ring**.

Step 2 Click the **Console** button on the **action ring** to open the node's **Console** pane at the bottom of the **Workbench**.

Figure 13: Open Console Node Action



Step 3 Click the **Open Console** button in the **Console** pane at the bottom of the **Workbench** to activate the console connection.

Stopping a Simulation

Lab simulations are not stopped automatically. A lab simulation will continue to run and consume resources even if you close the **Workbench** page for a lab or log out of the UI.

Step 1 First, ensure that no nodes are selected by clicking on an empty area of the **Workbench**.

Step 2 Select the **Simulate** tab at the bottom of the **Workbench** window.

Step 3 Click the **Stop** button.

CML will stop the simulation for this lab. The lab's nodes will be stopped, and the **Lab Manager** will no longer show this lab as running.

CHAPTER 4

External Connectivity for Simulations

The External Connector node (`ext-conn`) can be used to connect one or more simulation nodes to a network outside of the CML virtual environment. This connector can be configured as a **Bridge** or **NAT** device. To reduce complexity and overhead, the external connector node provides a single interface for connectivity.

External Connector - Bridge Mode

Bridge mode provides an unrestricted layer-2 connection to networks outside of the CML virtual environment. Standard networking rules apply. You must provide the required networking configuration on all simulation nodes that have been connected to the external network.

Caution It is possible for you to cause disruption on your real network or to trigger a loss of access to the CML server when you simulate a lab that uses bridge mode on an external connector. For example, ports on IOSv-L2 nodes are L2 by default with PVST STP and auto-negotiation for trunking enabled. If your topology has such an L2 device configured in switch-mode and if one of the switch ports is in the same broadcast domain as the external connector's port, when the L2 port comes up, it will transmit BPDUs that can trigger either an err-disable on an upstream switchport or a spanning tree event that can cause wider network disruption.

In this example, out-of-band (OOB) management connectivity is provided to multiple nodes using a single `ext-conn` node and leveraging an **Unmanaged Switch**.

Use Case	Provide layer-2 access to the simulation from the external network OOB (out of band) management of simulated devices.
Topology	**IOSv** router connecting an **IOSv-L2** switch
Required Nodes	**External Connector** *(×1)* **Unmanaged Switch** *(×1)* **IOSv** *(×1)* **IOSvL2***(×1)*

Step 1 Create a new lab in the **Lab Manager**.

Step 2 **Optional:** Give the new lab a name. Example: *bridge_connector*.

Step 3 Click the *bridge_connector* lab tile to open the **Workbench**.

Step 4 Drag-and-drop the required nodes onto the topology canvas.

Step 5 Connect **`ext-connB`** to **`unmanaged-switch`**.

Step 6 Connect the **`iosv`** and **`iosvl2`** nodes to **`unmanaged-switch`** using ***gi0/0*** to any port on the **`unmanaged-switch`**.

Step 7 Select the **`ext-conn`** node.

Step 8 Click the **Edit Config** tab in the bottom pane.

Step 9 Click **Bridge** in the **Edit Config** pane to select *bridge* mode.

Step 10 Press the `ESC` key to deselect the **`ext-conn`** node.

Step 11 **Optional:** Click the **Design** tab in the bottom pane.

Step 12 **Optional:** Click **Build Initial Bootstrap Configurations** in the **Design** pane.

This action provides a basic configuration and a system-assigned username and password of `cisco/cisco` to the Cisco routers in the lab.

Step 13 Click the **Simulate** tab in the bottom pane.

Step 14 Click **Start** in the **Simulate** pane.
The system will start a lab simulation, and the node VMs will start booting.

Step 15 Wait for all nodes to finish booting.

Once all nodes have booted, you still need to configure each node for OOB management. This example will show how to configure basic connectivity using a VRF for management and routing on interface `gi0/0`.

Step 16 Select the IOSv node.

Step 17 Click the **Console** tab in the bottom pane.

Step 18 Click the **Open Console** button in the **Console** pane.

Step 19 Log in.
Use the credentials that you set in the initial configs for the devices.

Username: **`cisco`**
Password: **`cisco`**

Step 20 Add a VRF config to the node.

Sample VRF config:

```
enable
conf t
vrf definition Mgmt-intf
address-family ipv4
int gi0/0
ip address n.n.n.n m.m.m.m (IP address and subnet mask)
exit
ip route vrf Mgmt-intf 0.0.0.0 0.0.0.0 g.g.g.g (source destination gateway)
```

Step 21 For each node that requires OOB management to the bridged network, add a VRF configuration by repeating the steps from Step 16, on page 22 to Step 20, on page 22.

External Connector - NAT Mode

NAT connectivity can be provided to any connected node in the topology using the External Connector node (`ext-conn`). This connection can be leveraged, for example, to install packages from a Linux repository or other external public or private package repository.

An IP address is dynamically assigned to the connected node using the next available address from the DHCP pool.

Table 1: NAT Network

Network	`192.168.255.0 /24`
Gateway	`192.168.255.1`
DHCP Pool	`192.168.255.2 - .254`

The router and switch used in the following topology example are not required to leverage NAT mode. They are included only for demonstration and to provide a use case reference.

Use Case	Provide NAT access to the external network for a Linux server running in a simulation.
Topology	**IOSv** router connecting an **IOSv-L2** switch and a **Server** node connected to an upstream switch and an **External Connector** node.
Required Nodes	**External Connector** *(×1)* **Server** *(×1)* **IOSv** *(×1)* **IOSvL2***(×1)*

Note The NAT network is not currently editable. This network is the internal interface of the External Connector and is not exposed to the **Workbench**. Making the NAT network for the External Connector node configurable via the **Workbench** is on the CML roadmap and is planned for a future release.

Step 1 Create a new lab in the **Lab Manager**.

Step 2 **Optional:** Give the new lab a name. Example: **`nat_connector`**.

Step 3 Click the *nat_connector* lab tile to open the **Workbench**.

Step 4 Drag-and-drop the required nodes onto the topology canvas.

Step 5 Connect **`ext-conn`** to **`server`**, using `eth1`.

Step 6 Connect the **`iosv`** and **`iosvl2`** nodes to each other using `gi0/1`.

Step 7 Connect the **`server`** and **`iosvl2`** nodes to each other using `eth0` to `gi1/0`.

Step 8 Select the **`ext-conn`** node.

Step 9 Click the **Edit Config** tab in the bottom pane.

Step 10 Click **NAT** in the **Edit Config** pane to select *NAT* mode.

Step 11 Press the `ESC` key to deselect the **`ext-conn`** node.

Step 12 **Optional:** Click the **Design** tab in the bottom pane.

Step 13 **Optional:** Click **Build Initial Bootstrap Configurations** in the **Design** pane.

This action provides a basic configuration and a system-assigned username and password of `cisco/cisco` to the Cisco routers in the lab.

Step 14 Click the **Simulate** tab in the bottom pane.

Step 15 Click **Start** in the **Simulate** pane.
The system will start a lab simulation, and the node VMs will start booting.

Step 16 Wait for all nodes to finish the boot process. Once the **server** node displays a solid green dot, it is then ready for use. Next, we can verify the NAT connectivity via the **External Connector**.

Step 17 Select the **server** node.

Step 18 Click the **Console** tab in the bottom pane.

Step 19 Click the **Open Console** button in the **Console** pane.

Step 20 Log in.

Use the credentials that you set in the initial configs for the devices.

Username: **`cisco`**
Password: **`cisco`**

Step 21 Run the following command to verify connectivity: `ifconfig eth1`
The command should return an IP address for subnet **`192.168.255.0/24`**.

Step 22 Run the following command to verify connectivity: `netstat -nr`
The command should return the default gateway IP of **`192.168.255`**.

CHAPTER 5

Breakout Tool

Breakout Tool Overview

The Breakout Tool gives you local access to consoles and graphical interfaces of VMs running in a remote lab. The telnet protocol is used for console access, and VNC protocol is used for graphics-capable VMs. The Breakout Tool is a single executable file that you run on the command line. It provides a kind of proxy connection from the local machine, where the tool has been configured and started, to the nodes in the lab simulation. Once you install it, you can configure the Breakout Tool using a web interface that is accessible via the localhost (i.e. `127.0.0.1`or `[::1]`) or loopback address on port `8080` by default. The port and listen address and the CML server's URL can be configured via command line options or the tool's configuration file (`config.yaml`). Using the Breakout Tool, you can use your favorite terminal emulator app to connect to your nodes' consoles on configurable local ports.

Installing the Breakout Tool

Step 1 Log into the CML server UI.

Step 2 On the **Lab Manager** page, click **Tools** > **Breakout Tool**.

Step 3 The **Breakout** page will open in a new window. The page includes instructions for using the Breakout Tool.

Step 4 Scroll to the bottom of page. There is a list of links to **`breakout`** executable binaries for a variety of platforms.

Step 5 Click on the appropriate **`breakout`** executable file for your local computer's operating system to download the file.

Step 6 Save the **`breakout`** binary to its own folder. The Breakout Tool will generate configuration and labs files to this folder.

Step 7 **(macOS and Linux)** Open a terminal application.

On macOS and Linux operating systems, you must make the binary executable before you can run it.

Step 8 **(macOS and Linux)** Change directory to the Breakout Tool's location: `cd /path/to/breakout_tool`

Step 9 **(macOS and Linux)** Change permissions to set the execution bit: `chmod u+x breakout-`*`platform`*`-x86_amd64`

You have downloaded and installed the Breakout Tool and are ready to run it on your local machine.

What to do next

You must configure the Breakout Tool before you can start using it with your labs.

Breakout Tool Command Line Interface

The Breakout Tool is provided as a single executable file. While the executable file has a platform-specific name, such as **`breakout-windows-x86_amd64.exe`**, this command reference refers to it as **`breakout`**. Open a local terminal or command prompt to run **`breakout`** on the command line: you must specify a command and may optionally provide one or more option flags.

breakout *[flags]* COMMAND

Command	Description
config	Creates a default configuration file.
init	Retrieves lab information from the CML server. When a lab ID or unique lab name is provided as a second argument, the Breakout Tool will only retrieve the information for the specified lab and enable it by default.
run	Runs the tool, reading all configuration information from config files, CLI flags, and environment variables.
ui	Runs the tool with a web interface for both configuring the tool and enabling connections.

Option Flag	Option Argument	Description
-noverify	N/A	Overrides the `verify` setting in the configuration. When specified, **`breakout`** does not verify the TLS certificates.
-listen	`ipaddress`	The `address` overrides the listen address. This option can be used to specify, e.g. `-listen 127.0.0.1`, that **`breakout`** should listen on the localhost IPv4 address instead of the IPv6 in the default configuration.
-labs	`filename`	Specifies the name of the lab configuration file to use (default: labs.yaml).
-config	`filename`	Specifies the name of the global configuration file to use (default: config.yaml).
-extralf	N/A	Sends an extra LF when serial line is opened.
-port	`num`	Specifies the local port number to listen on for UI mode (default: `8080`).

Breakout Tool Configuration Settings

Other than the controller address, username, and password, the following settings for the Breakout Tool may not be empty, but for most applications you may simply use the default values. The Breakout Tool's `config` command and the **Breakout UI** can both write a default config.yaml with default configuration values for these settings.

Setting	Description
Controller Address	The URL for the CML server.
Username	The CML application user; normally, this setting is just the owner of the lab.
Password	The application password for the specified user.
Console Start Port	Connections to console ports will be enumerated starting with the specified port number.
VNC Start Port	Connections to VNC consoles will be enumerated starting with the specified port number.
Listen Address	Local IP address or interface used by the **Breakout UI** web interface when it is run with the Breakout Tool's **`ui`** command.
UI Server Port	The local TCP port used by the **Breakout UI** web interface when it is run with the Breakout Tool's **`ui`** command.
Labs Filename	The **Breakout UI** uses the **Labs Filename** (`labs.yaml`) to save the settings for all running labs from the CML server. This file is *not* the same as the individual lab topology files shown in the **Lab Manager**. Settings for all running labs and running (default) nodes will be written to this file when you click **Save labs to local disk** in the **Breakout UI**.

Configuring the Breakout Tool

The Breakout Tool may be run entirely from the command line, specifying all settings using the tool's command line options. For most users, we recommend that you configure the Breakout Tool with the provided web interface. The following section describes how to configure the tool for the first time and save all settings for future use.

Before you begin

The Breakout Tool is already installed on the local machine. You have opened a local terminal application to the Breakout Tool's installation directory.

Step 1 In the **Lab Manager** or the **Workbench**, start one or more labs.

Step 2 Run the Breakout Tool and specify the `ui` command.

Example:

The exact command will depend on your platform. For example, on Windows, it would be:

```
breakout-windows-x86_amd64 ui
```

On Mac OS X, it would be:

```
./breakout-macos-x86_amd64 ui
```

Step 3 Wait until the Breakout Tool's web UI is ready and the command outputs a message indicating that it is running and "serving UI/API".

On first launch, the tool will try to connect to an unspecified host, and you must wait until the connection times out.

Example:

Figure 14: CLI output that indicates the Breakout Tool's web UI is ready

```
Starting up...
W0219 18:13:43.020647   10128 run.go:238] open labs.yaml: The system cannot find the file specified.
Running... Serving UI/API on http://[::1]:8080, Ctrl-C to stop.
```

Step 4 With web UI running, open a supported web browser and navigate to the Breakout Tool's configuration page, http://localhost:8080 by default.
The **Breakout UI** will be shown in the browser.

Step 5 Click **Configuration** at the top of the **Breakout UI**.

Step 6 Enter the IP Address or hostname of your CML installation in the **Controller Address** field.

Step 7 Enter the username and password that you use to log into the CML web interface in the **Username** and **Password** fields.

The Breakout Tool will only have access to simulations that the specified username can access. On a multi-user system, if you want to connect to a particular lab simulation, be sure to use the credentials of a user who can see that lab in the **Lab Manager**.

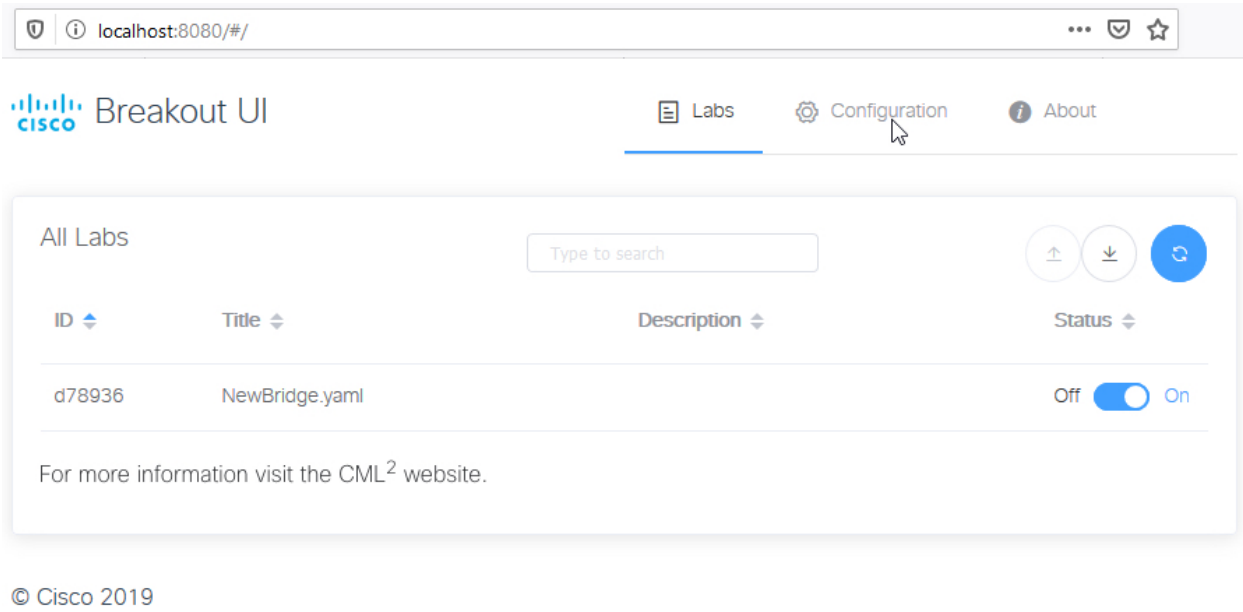
Step 8 Click the **Save** button.
The configuration settings are saved to the `config.yaml` file in the Breakout Tool's installation directory.

Step 9 Click **Labs** at the top of the **Breakout UI**.

Step 10 After changing the configuration, you may need to click the **Refresh** button at the top of the **All Labs** page to refresh the list of labs.

Example:

Figure 15: Breakout UI's All Labs page



All running labs are displayed on the **Labs** page of the **Breakout UI**.

Since the Breakout Tool's configuration settings are stored to the `config.yaml` file, they will be used the next time you run the Breakout UI. You shouldn't have to provide the **controller address** or user credentials on the **Configuration** page of the Breakout UI in the future unless those values change.

What to do next

You're now ready to use the Breakout Tool to connect to nodes in your simulation.

Using the Breakout Tool

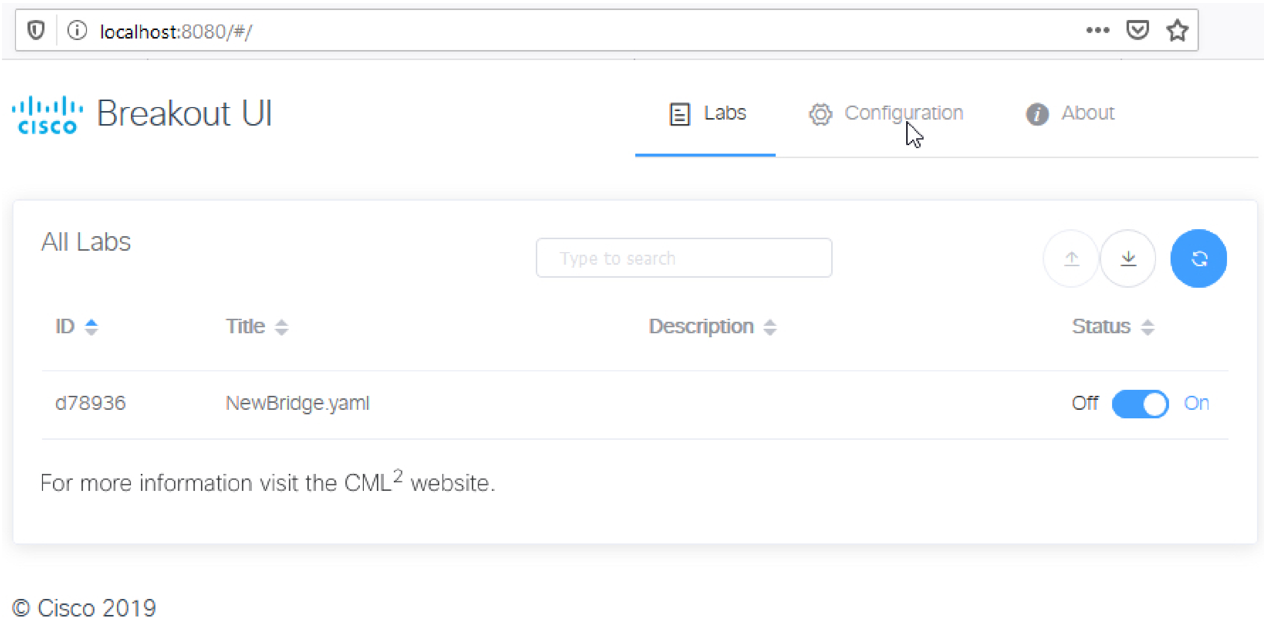
Before you begin

With one or more lab simulations running on the CML server, start the **Breakout UI** locally, using the configured settings from the config.yaml file. See Configuring the Breakout Tool, on page 27.

Step 1 Open the **Breakout UI** in a supported web browser.

Example:

Figure 16: Breakout UI's All Labs page

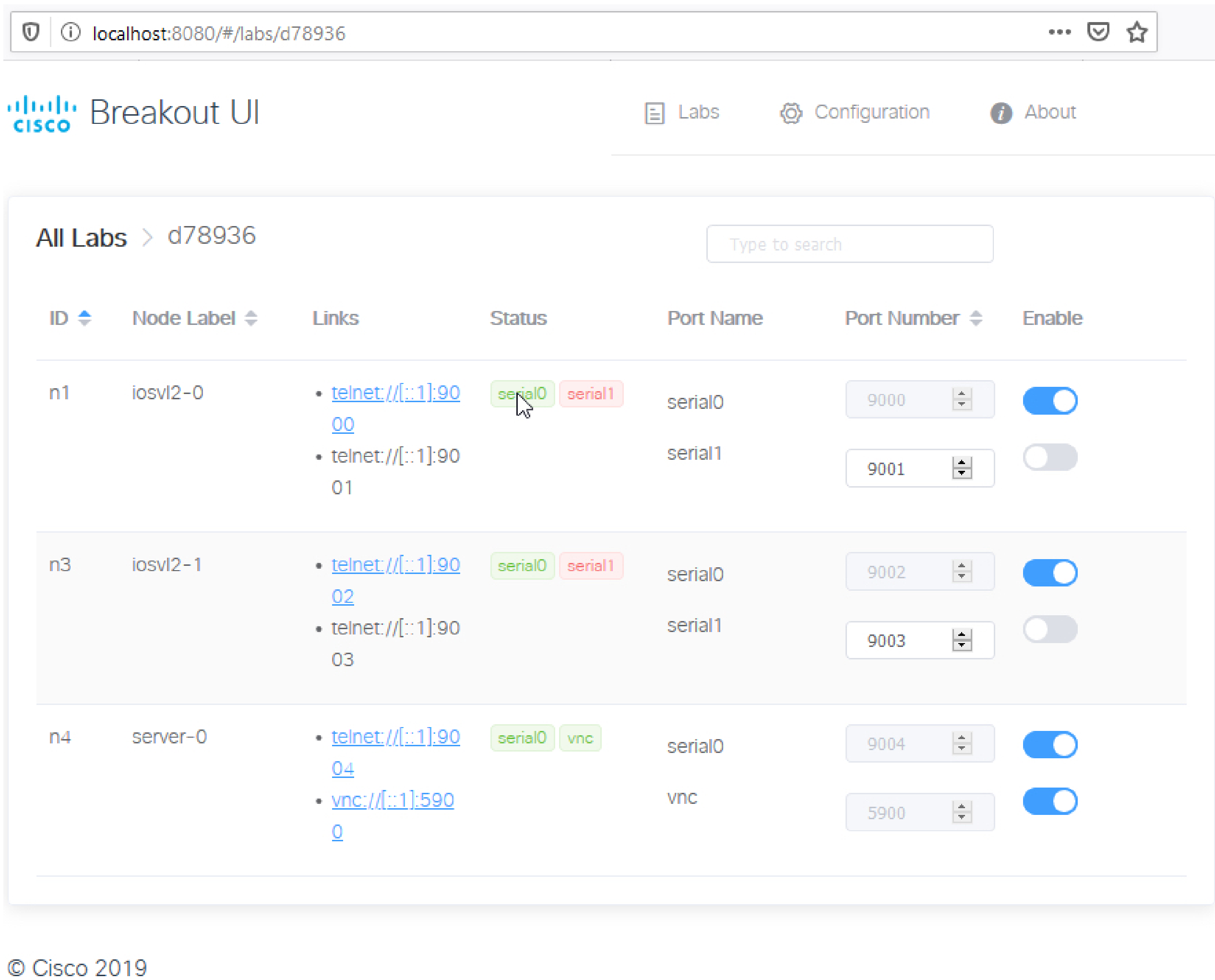


The **Breakout UI** shows your list of running labs in the **Labs** page.

Step 2 Click on a lab in the **All Labs** list to view the available serial and VNC ports for the nodes in that lab simulation.

Example:

Figure 17: The Breakout UI's page for a specific lab



Step 3 **Optional:** You may click the **Enable** slider to enable or disable specific node and port name connections.

The Breakout Tool will only provide local access to enabled nodes and port names. Enabled serial and VNC connections will be displayed in green under the **Status** column.

Step 4 **Optional:** You may set the local port number for specific node and port name combinations. To change the port number, you may need to click the **Enable** slider to disable that connection first.

Step 5 To open an enabled telnet or VNC connection, click on the link in the **Links** column. Alternatively, you may simply start a local terminal emulator application or VNC client and then connect to localhost on the specified port number.

For the links to function as expected, you must configure a telnet:// and vnc:// connection handler in your web browser.

CHAPTER 6

Custom VM Images

CML is bundled with a set of node definitions, such as ASAv and IOSv, and a single VM image for each node definition. CML permits you to upload additional VM images (as `qcow2` files) for use in your labs. After you upload a new `qcow2` file, you must create an image definition for the new VM image. For example, you could add an alternate version of ASAv to your CML server to match the ASAv version used in a real network that you are modeling in CML. The system would then have two image definitions associated with the ASAv node definition. When you use an ASAv node in your CML lab, you could then choose which of the ASAv image definitions to use for that node.

In addition to uploading alternate versions of existing VM images, the custom image feature permits you to upload third-party VM images, such as a Windows VM, that are not bundled by default. You are responsible for obtaining and properly licensing the third-party VM image. In addition to creating an image definition for the third-party VM image, you may also need to create a new node definition. While we do not provide support for VM images that are not included with the product, if the third-party VM runs on qemu-kvm on CentOS Linux, then CML can generally start the VM and link it to the other nodes in the lab topology. This chapter shows how to load a custom image into the CML server and use it in your labs.

Preparing Custom Images

Step 1 From the **Lab Manager**; select **Tools** > **Node and Image Definitions**.
The UI opens the **Node and Image Definitions** page.

Step 2 Click **Manage Uploaded Images**.

Step 3 Upload and select a VM image file.

Click **Browse** and locate a local VM image file to upload. Click the **Upload Image** button.

Alternatively, upload an image to the CML server via scp.

New images are displayed under the **Uploaded Images** section. Click **Refresh** to update the list.

Step 4 Click **Create New Image Definition**

Step 5 Enter values for the image definition's fields.

ID
Each image on the system must have a unique ID.

Label
The label appears in the node's image selection list in the **Workbench** and generally indicates the VM image's OS name and version, such as `IOSv 15.6(3)`.
Description
This field can provide a longer description of the image.
Disk Image
Select the image that you uploaded from the list.
Node Definition
Select the appropriate node definition to be associated with selected image.

Step 6 Click **Create Image Definition**.

A new image is created with the image properties from the form. It is now available for use in your labs.

Using Custom Images

Once you have added the custom image to your system, you can use it with any node that uses the same node definition that is associated with the custom image definition. See also Preparing Custom Images, on page 33.

Step 1 Drag-and-drop a node to the topology canvas in the **Workbench**. Use a node definition that has at least one custom image associated with it.

Step 2 Select the node.

Step 3 Click the **Simulate** tab in the bottom pane.

Step 4 Select the custom image from the **Image Definition** drop-down list in the **Simulate** pane.

When you start the lab simulation, this node will use the custom image instead of the default image for the associated node definition.