The CloudLab Manual

Version 2015-04-08 (a876d4f)

The CloudLab Team

Wednesday, April 8th, 2015

CloudLab is up and running, but in active development. For more information on the deployment schedule for the full version of CloudLab, see the CloudLab website. Please also see the "Status Notes" chapter of this document for notes regarding the status of the facility.

The HTML version of this manual is available at http://docs.cloudlab.us/

CloudLab is a "meta-cloud"—that is, it is not a cloud itself; rather, it is a facility for building clouds. It provides bare-metal access and control over a substantial set of computing, storage, and networking resources; on top of this platform, users can install standard cloud software stacks, modify them, or create entirely new ones.

The initial CloudLab deployment will consist of approximately 15,000 cores distributed across three sites at the University of Wisconsin, Clemson University, and the University of Utah. CloudLab will interoperate with existing testbeds including GENI and Emulab, to take advantage of hardware at dozens of sites around the world.

The control software for CloudLab is open source, and is built on the foundation established for Emulab, GENI, and Apt. Pointers to the details of this control system can be found on CloudLab's technology page.

Take a look at the status notes, and then get started!

Contents

1 CloudLab Status Notes

CloudLab is currently open. Both the hardware and software are being deployed in stages. Our goal is to get community feedback all along to help guide further deployments and development. So, please keep in contact with us (see the "getting help" chapter), let us know what is (or isn't) working for you, and what features you'd like to see in CloudLab.

There are a few things to note:

- The hardware is being **brought up in stages**; see `https://www.cloudlab.us/hardware.php` for a description of the eventual CloudLab hardware, and `https://www.cloudlab.us/#availability` for a schedule for availability. See the hardware chapter for a description of the clusters that are available so far.
- We have only built a **single cloud-specific profile** so far, a small installation of OpenStack. Experimentation on CloudLab is *not* limited to OpenStack, nor to this profile; it is merely provided as an example. There are several existing methods for users to create custom profiles, and several more are planned. See the sections of this manual on profiles and creating profiles for more details.
- There are many planned features that are **not yet implemented**.
- This documentation is **incomplete and evolving**.

With all of this in mind, give it a try!

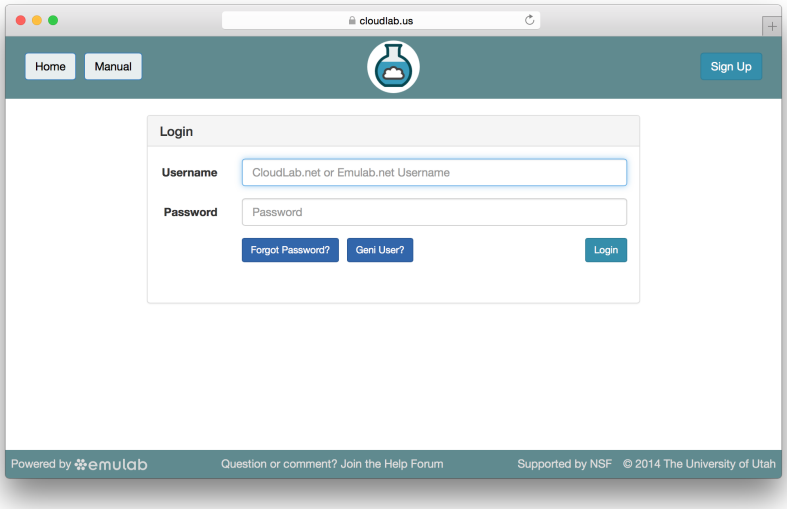
2 Getting Started

This chapter will walk you through a simple experiment on CloudLab and introduce you to some of its basic concepts.

Start by pointing your browser at `https://www.cloudlab.us/`.

1. **Log in**

   You'll need an account to use CloudLab. If you already have an account on Emulab.net, you may use that username and password. Or, if you have an account at the GENI portal, you may use the "GENI User" button to log in using that account. If not, you can apply to start a new project at `https://www.cloudlab.us/signup.php` and taking the "Start New Project" option. See the chapter about CloudLab users for more details about user accounts.



2. **Select a profile**

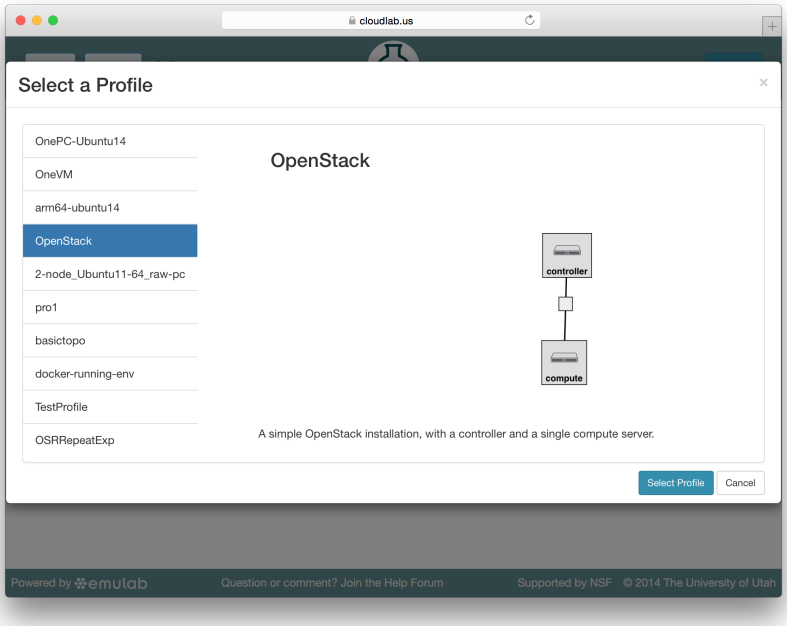
   Clicking the "Change Profile" button will let you select the profile that your experiment will be built from. A profile describes a set of resources (both hardware and software) that will be used to start your experiment. On the hardware side, the profile will control whether you get virtual machines or physical ones, how many there are, and what the network between them looks like. On the software side, the profile specifies the operating system and installed software.

   Profiles come from two sources. Some of them are provided by CloudLab itself, and provide standard installation of popular operating systems, software stacks, etc.

Others are created by other researchers and may contain research software, artifacts and data used to gather published results, etc. Profiles represent a powerful way to enable repeatable research.

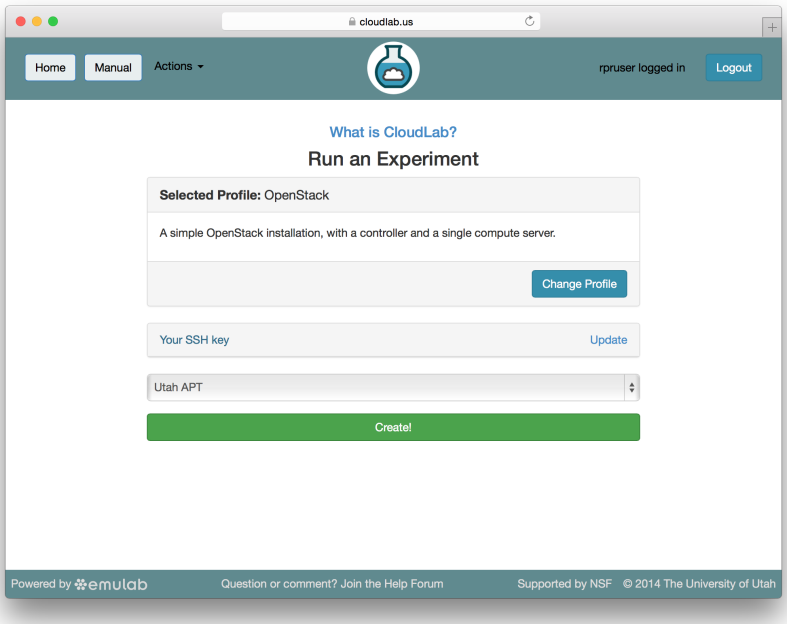
The large display in this dialog box shows the network topology of the profile, and a short description sits below the topology view.

The `OpenStack` profile that we've selected here will give you a small OpenStack installation with one master node and one compute node. It provides a simple example of how complex software stacks can be packaged up within CloudLab. If you'd prefer to start from bare metal, look for one of the profiles that installs a stock operating system on physical machines.



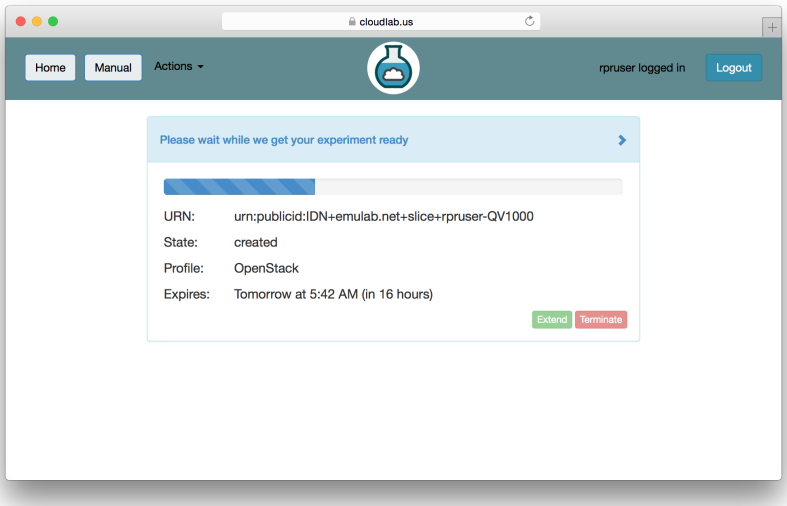
3. **Pick a cluster**

CloudLab can instantiate profiles on several different backend clusters. The cluster selector is located right above the "Create" button; the the cluster most suited to the profile you've chosen will be selected by default.



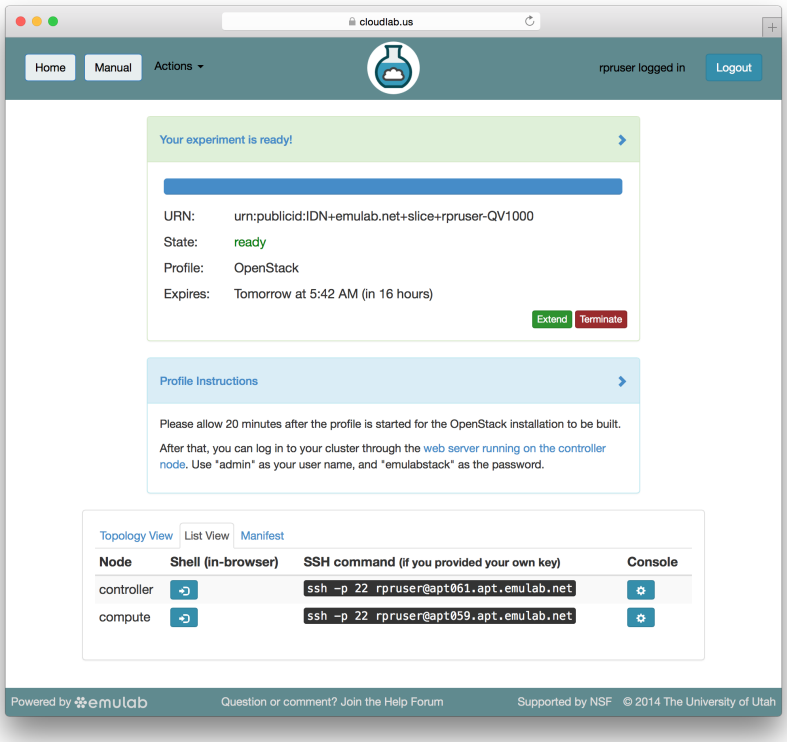
4. **Click Create!**

   When you click the "Create" button, CloudLab will start preparing your experiment by selecting nodes, installing software, etc. as described in the profile. What's going on behind the scenes is that on one (or more) of the machines in one of the CloudLab clusters, a disk is being imaged, a set of physical machines booted, accounts created for you, etc. This process usually takes a couple of minutes.



5. **Use your experiment**

   When your experiment is ready to use, the progress bar will be complete, and you'll be given a lot of new options at the bottom of the screen.

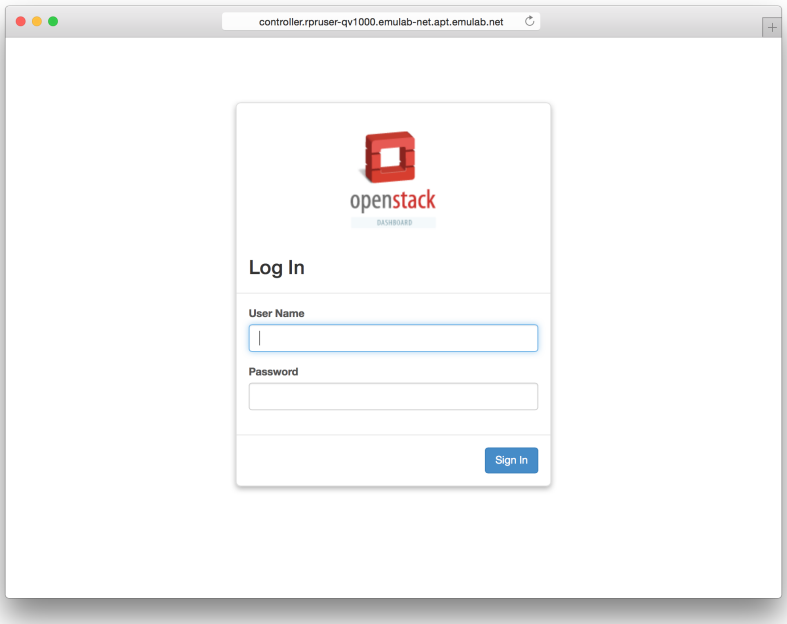


The "Topology View" shows the network topology of your experiment (which may be as simple as a single node). Clicking on a node in this view brings up a terminal in your browser that gives you a shell on the node. The "List View" lists all nodes in the topology, and in addition to the in-browser shell, gives you the command to `ssh` login to the node (if you provided a public key). The "Manifest" tab shows you the technical details of the resources allocated to your experiment. Any open terminals you have to the nodes show up as tabs on this page.

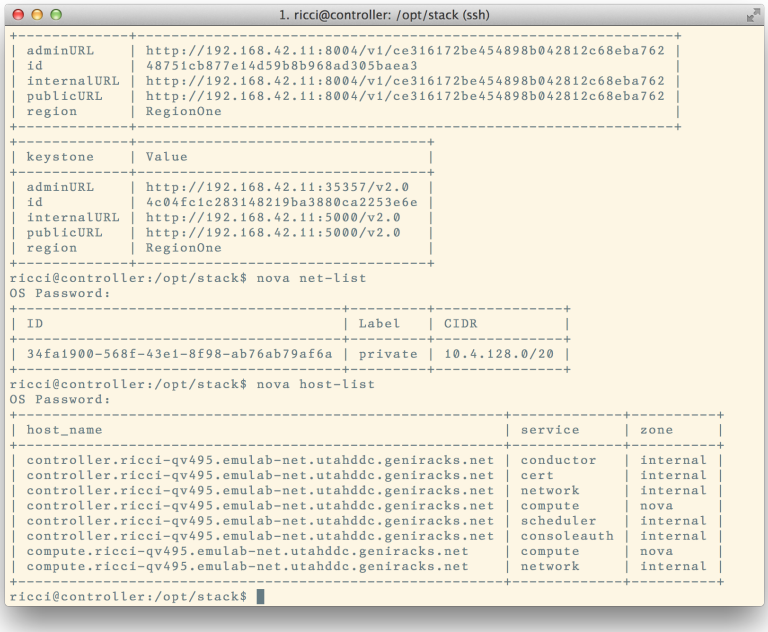
Clicking on the "Profile Instructions" link (if present) will show instructions provided by the profile's creator regarding its use.

Your experiment is yours alone, and you have full "root" access (via the `sudo` command). No one else has access to the nodes in your experiment, and you may do anything at all inside of it, up to and including making radical changes to the operating system itself. We'll clean it all up when you're done!

If you used our default OpenStack profile, the instructions will contain a link to the OpenStack web interface. The instructions will also give you a username and password to use.



Since you gave CloudLab an `ssh` public key as part of account creation, you can log in using the `ssh` client on your laptop or desktop. The contoller node is a good place to start, since you can poke around with the OpenStack admin commands. Go to the "list view" on the experiment page to get a full command line for the `ssh` command.



Your experiment will **terminate automatically after a few hours.** When the experiment terminates, you will **lose anything on disk** on the nodes, so be sure to copy off anything important early and often. You can use the "Extend" button to submit a request to hold it longer, or the "Terminate" button to end it early.

2.1 Next Steps

- Read the notes on the status of CloudLab
- Try a profile that runs bare metal and set up a cloud stack yourself
- Making your own profiles is easy: see the chapter on profile creation for instructions.
- If you need help, or have questions or comments about CloudLab's features, contact us!

3 CloudLab Users

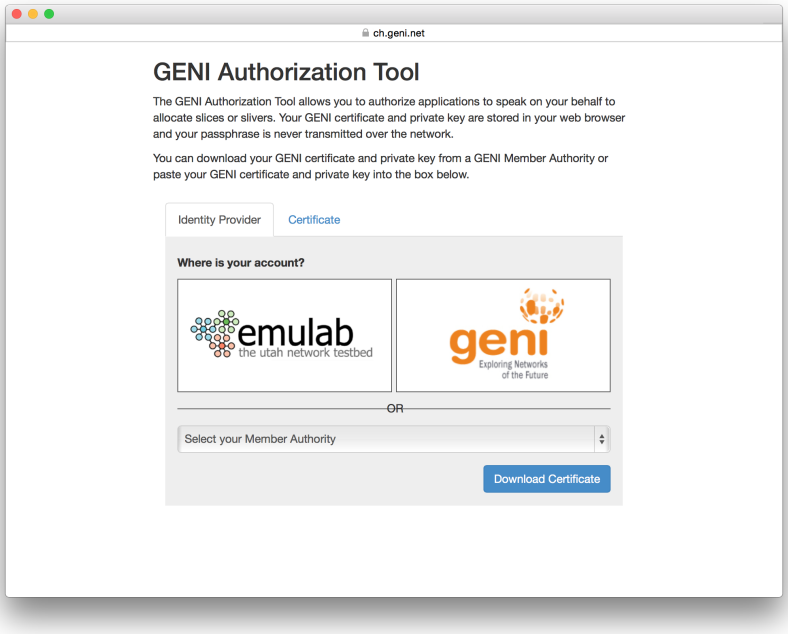
3.1 Registered Users

Registering for an account is quick and easy. Registering doesn't cost anything, it's simply for accountability. We just ask that if you're going to use CloudLab for anything other than light use, you tell us a bit more about who you are and what you want to use CloudLab for.

Users in CloudLab are grouped into *projects*: a project is a (loosely-defined) group of people working together on some common goal, whether that be a research project, a class, etc. CloudLab places a lot of trust on project leaders, including the ability to authorize others to use the CloudLab. We therefore require that project leaders be faculty, senior research staff, or others who are relatively senior positions.

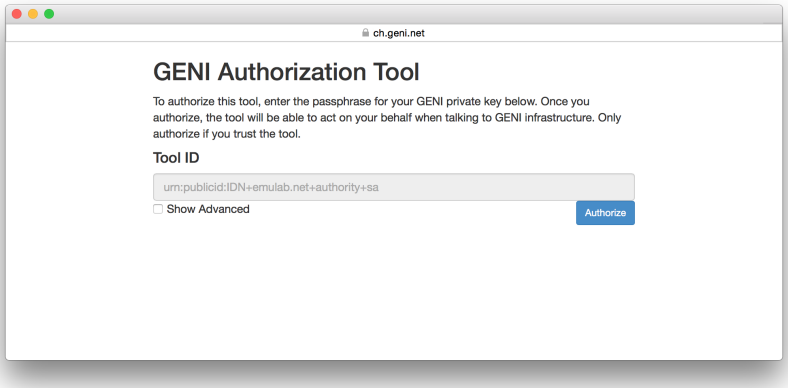
3.2 GENI Users

If you already have a GENI account, you may use it instead of creating a new CloudLab account. On the login page, select the "GENI User" button. You will be taken to a page like the one below to select where you normally log into your GENI account.



From here, you will be taken to the login page of your GENI federate; for example, the login page for the GENI portal is shown below.

After you log in, you will asked to authorize the CloudLab portal to use this account on your behalf. If your certificate at your GENI aggregate has a passphrase on it, you may be asked to enter that passphrase; if not, (as is the case with the GENI portal) you will simply see an "authorize" button as below:



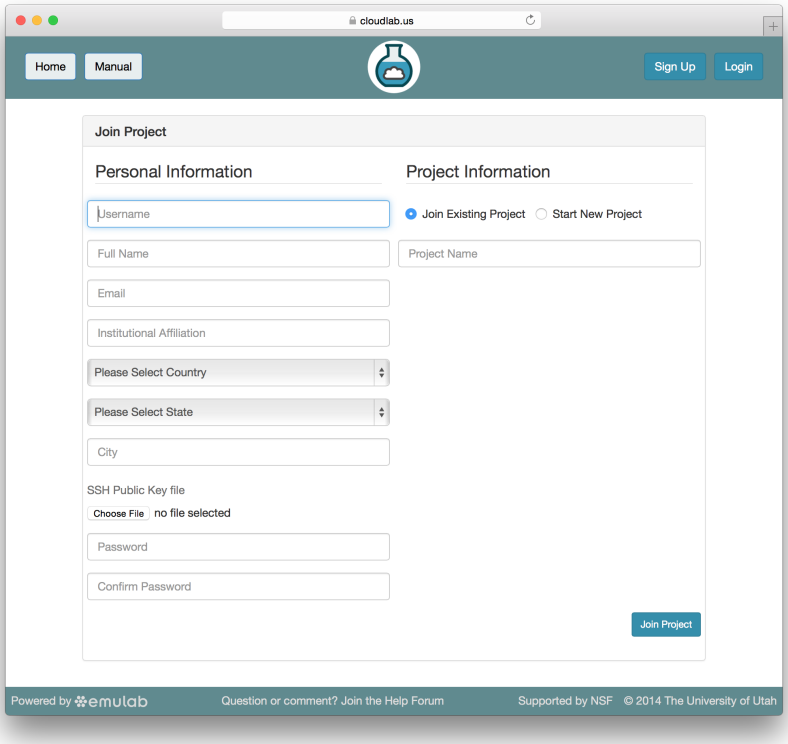
That's it! When you log in a second time, some of these steps may be skipped, as your browser has them cached.

3.3 Register for an Account

To get an account on CloudLab, you either join an existing project or create a new one. In general, if you are a student, you should join a project led by a faculty member with whom you're working.

If you already have an account on Emulab.net, you don't need to sign up for a new account on CloudLab—simply log in with your Emulab username and password.

3.3.1 Join an existing project



To join an existing project, simply use the "Sign Up" button found on every CloudLab page. The form will ask you a few basic questions about yourself and the institution you're affiliated with.

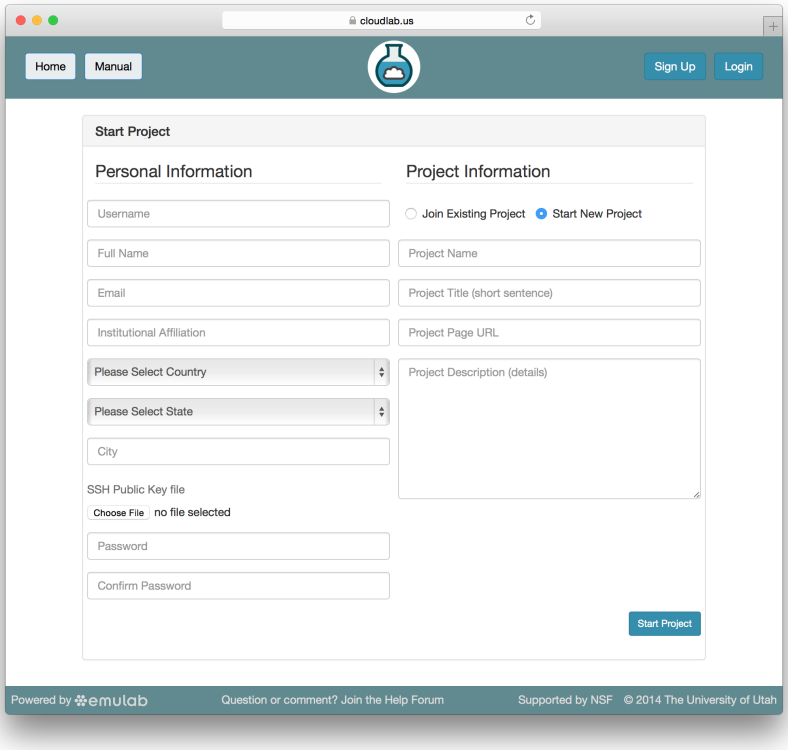
An SSH public key is required; if you're unfamiliar with creating and using ssh keypairs, we recommend taking a look at the first few steps in GitHub's guide to generating SSH keys.

(Obviously, the steps about how to upload the keypair into GitHub don't apply to CloudLab.)

You'll be asked to enter the project ID for the project you are asking to join; you should get this from the leader of the project, likely your advisor or your class instructor. (If they don't already have a project on CloudLab, you can ask them to create one.) The leader of your project is responsible for approving your account.

CloudLab will send you email to confirm your address—watch for it (it might end up in your spam folder), as your request won't be processed until you've confirmed your address.

3.3.2 Create a new project



To start a new project, use the "Sign Up" button found on every CloudLab page. In addition to basic information about yourself, the form will ask you a few questions about how you intend to use CloudLab. The application will be reviewed by our staff, so please provide enough information for us to understand the research or educational value of your project. The review process may take a few days, and you will receive mail informing you of the outcome.

You should only start a new project if you are a faculty member, senior research staff, or in some other senior position. Students should ask their advisor or course instructor to create a new project.

Every person working in your project needs to have their own account. You get to approve

these additional users yourself (you will receive email when anyone applies to join.) It is common, for example, for a faculty member to create a project which is primarily used by his or her students, who are the ones who run experiments. We still require that the project leader be the faculty member, as we require that there is someone in a position of authority we can contact if there are questions about the activities of the project.

Note that projects in CloudLab are publicly-listed: a page that allows users to see a list of all projects and search through them does not exist yet, but it will in the future.

4 CloudLab and Repeatable Research

One of CloudLab's key goals is to enable *repeatable research*—we aim to make it easier for researchers to get the same software and hardware environment so that they can repeat or build upon each others' work.

CloudLab is designed as a *scientific instrument*. It gives full visibility into every aspect of the facility, and it's designed to minimize the impact that simultaneous slices have on each other. This means that researchers using CloudLab can fully understand why their systems behave the way they do, and can have confidence that the results that they gather are not artifacts of competition for shared hardware resources. CloudLab profiles can also be published, giving other researchers the exact same environment—hardware and software—on which to repeat experiments and compare results.

CloudLab gives exclusive access to compute resources to one experiment at a time. (That experiment may involve re-exporting those resources to other users, for example, by running cloud services.) Storage resources attached to them (eg. local disk) are also used by a single experiment at a time. See the planed features chapter for a discussion of exclusive access to switches.

5 Creating Profiles

In CloudLab, a profile captures an entire cloud environment—the software needed to run a particular cloud, plus a description of the hardware (including network topology) that the cloud software stack should run on.

When you create a new profile, you are creating a new RSpec, and, usually, creating one or more disk images that are referenced by that RSpec. When someone uses your profile, they will get their own experiment that boots up the resources (virtual or physical) described by the RSpec.

5.1 Creating a profile from an existing one

The easiest way to create a new profile is by cloning an existing one and customizing it to your needs. The basic steps are:

1. Find an existing profile that's closest to your needs
2. Create an experiment using that profile
3. Log into the node(s) in your experiment to install your software, configure the operating system, etc.
4. "Clone" the experiment to create a new profile

5.1.1 Preparation and precautions

To create profiles, you need to be a registered user.

Creating a profile can take a while, so we recommend that you extend your experiment while creating it, and contact us if you are worried your experiment might expire before you're done creating your profile. We also strongly recommend testing your profile fully before terminating the experiment you're creating it from.

Your home directory is **not** included in the disk image snapshot! You will need to install your code and data elsewhere in the image. We recommend `/local/`. Keep in mind that others who use your profile are going to have their own accounts, so make sure that nothing in your image makes assumptions about the username, home directory, etc. of the user running it.

Be aware the only the contents of disk (not running process, etc.) are stored as part of the profile, and as part of the creation process, your node(s) will be rebooted in order to take consistent snapshots of the disk.

For the time being, this process only works for single-node profiles; we will add support for multi-node profiles in the future.

5.1.2 Creating the Profile

1. **Create an experiment**

   Create an experiment using the profile that is most similar to the one you want to build. Usually, this will be one of our facility-provided profiles with a generic installation of Linux.

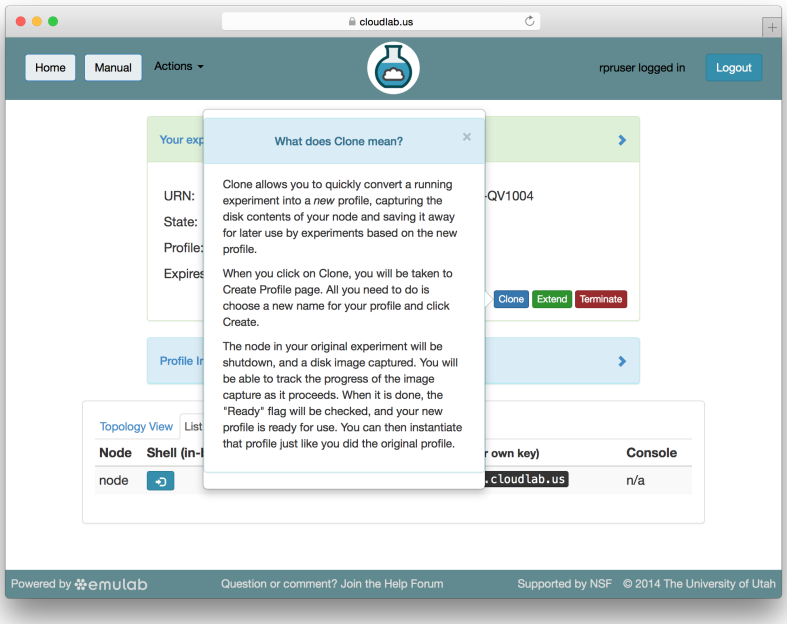
2. **Set up the node the way you want it**

   Log into the node and install your software, datasets, packages, etc. Note the caveat above that it needs to be installed somewhere outside of your home directory, and should not be tied to your user account.

3. **Clone the experiment to create a new profile**

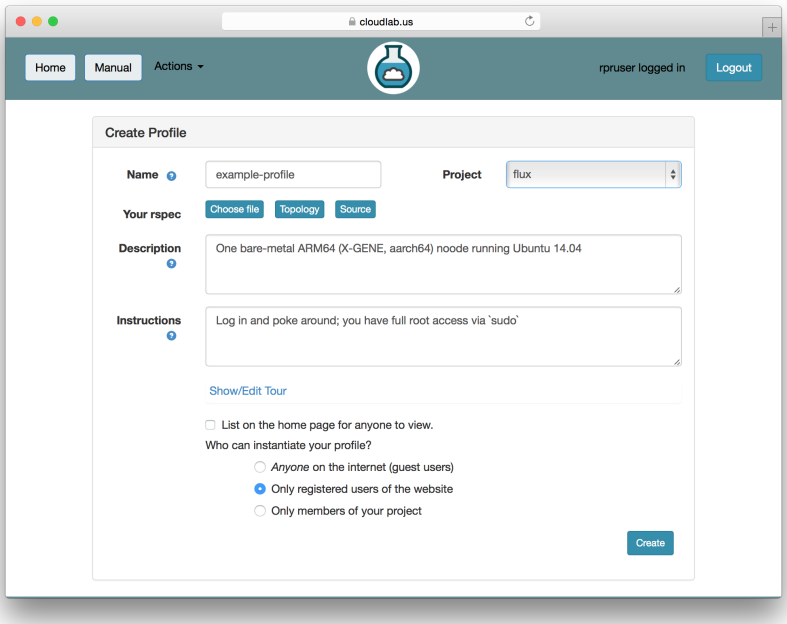
   While you are logged in, the experiment page for your active experiments will have a "clone" button. Clicking this button will create a new profile based on your running experiment.

   Specifically, the button creates a copy of the RSpec used to create the experiment, passes it to the form used to create new profiles, and helps you create a disk image from your running experiment.



4. **Fill out information for the new profile**

   After clicking on the “clone” button, you will see a form that allows you to view and edit the basic information associated with your profile.



Each profile must be associated with a project. If you're a member of more than one project, you'll need to select which one you want the profile to belong to.

Make sure to edit the profile's Description and Instructions.

The "Description" is the text that users will see when your profile is listed in CloudLab, when the user is selecting which profile to use. It is also displayed when following a direct link to your profile. It should give the reader a brief description of what they will get if they create an experiment with this profile. If the profile is associated with a paper or other publication, this is a good place to mention that. Markdown markup, including hyperlinks, are allowed in the profile description.

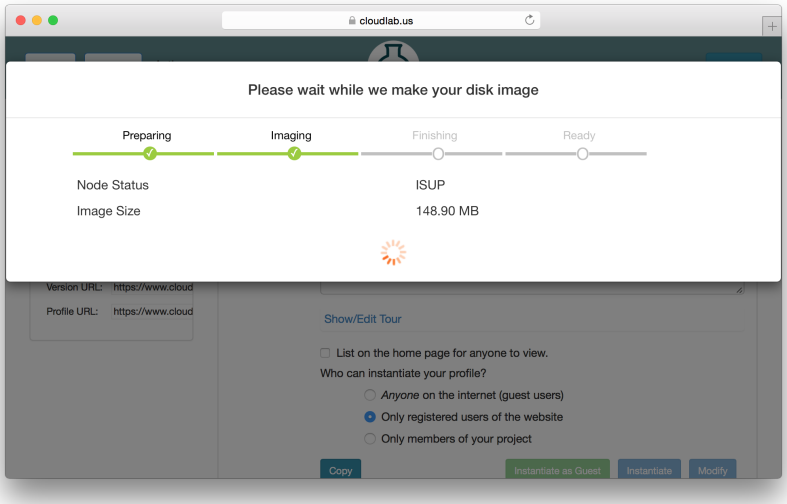
The "Instructions" text is displayed on the experiment page after the user has created an experiment using the profile. This is a good place to tell them where the code and data can be found, what scripts they might want to run, etc. Again, Markdown is allowed.

The "Steps" section allows you to create a tour of your profile, which is displayed after a user creates an experiment with it. This feature is mostly useful if your profile contains more than one node, and you wish to explain to the user what the purpose of each node is.

You have the option of making your profile usable to anyone, only registered CloudLab users, or members of your project. Regardless of the setting you chose here, CloudLab will also give you a direct link that you can use to share your profile with others of your choosing.

5. **Click "Create"**

   When you click the "Create" button, your node will be rebooted, so that we can take a consistent snapshot of the disk contents. This process can take several minutes or longer, depending on the size of your disk image. You can watch the progress on this page. When the progress bar reaches the "Ready" stage, your new profile is ready! It will now show up in your "My Profiles" list.



6. **Test your profile**

   Before terminating your experiment (or letting it expire), we strongly recommend testing out the profile. If you elected to make it publicly visible, it will be listed in the profile selection dialog on the front page of `https://www.cloudlab.us/`. If not, you can instantiate it from the listing in your "My Profiles" page. If the profile will be used by guest users, we recommend testing it as one yourself: log out, and instantiate it using a different username (you will also have to use an alternate email address.

7. **Share your profile**

   Now that your profile is working, you can share it with others by sending them direct links, putting links on your webpage or in papers, etc. See "§5.5 "Sharing Profiles"" for more details.

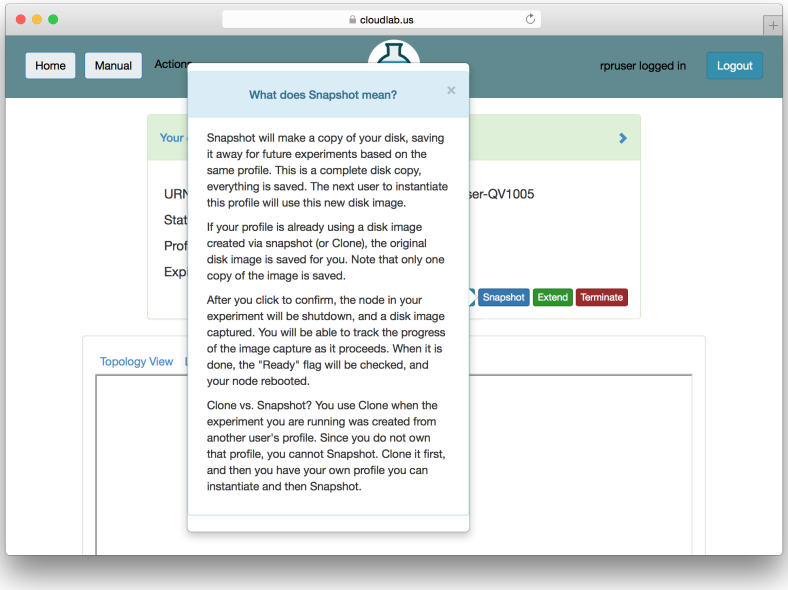
5.1.3 Updating a profile

There are planned features relating to this section: see "§9.1 “Versioned Profiles”" for more details.

You can update the metadata associated with a profile at any time by going to the “My Profiles” page and clicking on the name of the profile to go to the profile page. On this page, you can edit any of the text fields (Description, Instructions, etc.), change the permissions, etc.

As with cloning a profile, this snapshot feature currently only works with single-node profiles.

If you need to update the contents of the disk image in the profile, simply create a new experiment from the profile. (You will only see this button on experiments created from profiles that you own.) Once your experiment is ready, you will see a “Snapshot” button on the experiment page. Log into your node, get the disk changed the way you want, and click the button.

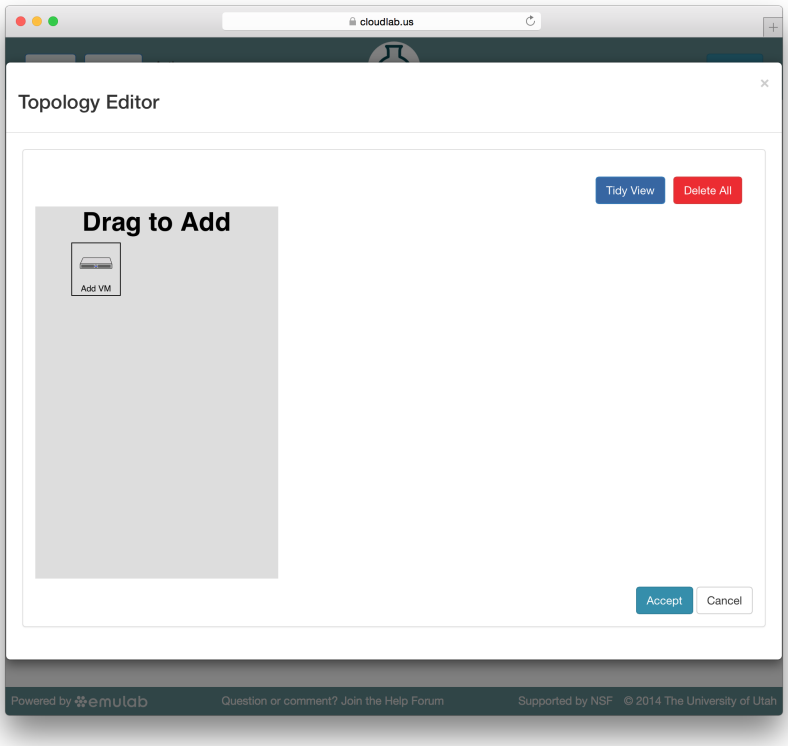


This button kicks off the same image creation process that occurs during cloning a profile. Once it’s finished, any new experiments created from the profile will use the new image.

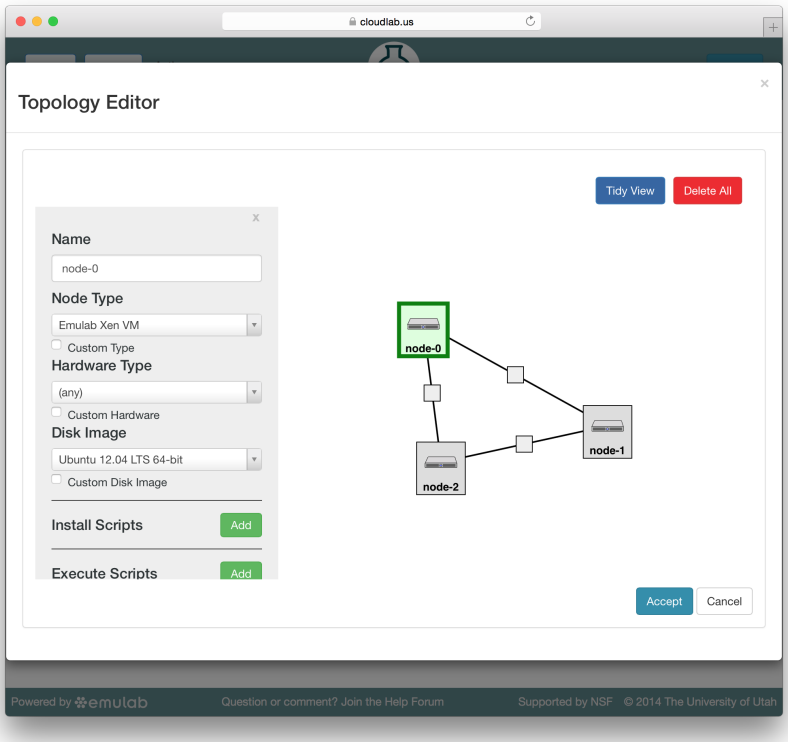
As with creating a new profile, we recommend testing the profile before letting your experiment expire. If something goes wrong, we do keep one previous image file for each profile; currently, the only way to get access to this backup is to contact us.

5.2 Creating a profile with a GUI

CloudLab embeds the Jacks GUI for simple creation of small profiles. Jacks can be accessed by clicking the "topology" button on the profile creation or editing page. Jacks is designed to be simple, and to ensure that the topologies drawn can be instantiated on the hardware available. Thus, after making certain choices (such as picking an operating system image) you may find that other choices (such as the node type) become limited.



Jacks has a "palette" on the left side, giving the set of node types (such as physical or virtual machines) that are available. Dragging a node from this palette onto the larger canvas area on the right adds it to the topology. To create a link between nodes, move the mouse near the first node, and a small black line will appear. Click and drag to the second node to complete the link. To create a LAN (multi-endpoint link), create a link between two nodes, then drag links from other nodes to the small grey box that appears in the middle of the original link.

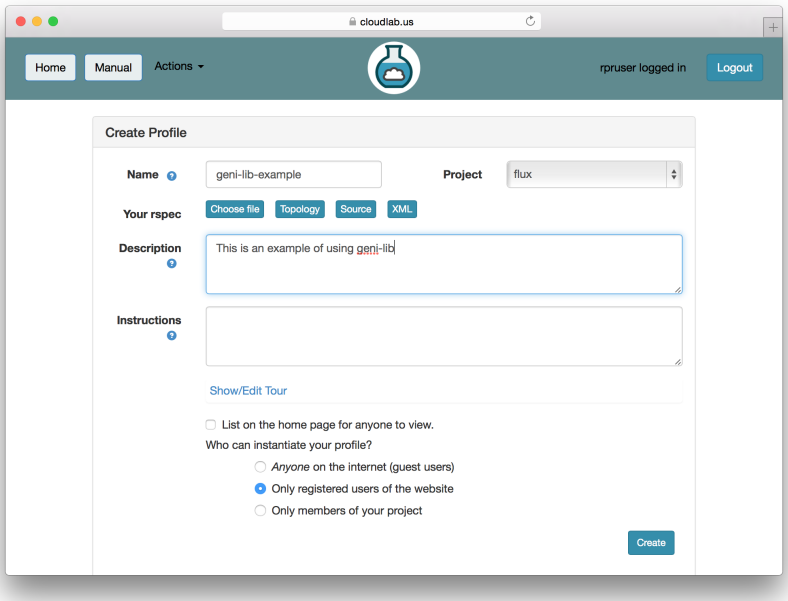


To edit the properties of a node or link, select it by clicking on its icon on the canvas. The panel on the left side will be replace by a property editor that will allow you to to set the disk image for the node, set commands to be run when the node boots, etc. To unselect the current node or link, and return to the palette on the left, simply click a blank area of the canvas.

5.3 Describing a profile with python and `geni-lib`

This feature is currently in **alpha testing** and may change in the future.

`geni-lib` is a tool that allows users to generate RSpec files from Python code. An **experimental** feature of CloudLab is the ability to use `geni-lib` scripts as the definition of a profile, rather then the more primitive RSpec format. When you supply a `geni-lib` script on the Create Profile page, your script is uploaded to the server so that it can be executed in the `geni-lib` environment. This allows the script to be verified for correctness, and also produces the equivalent RSpec representation that you can view if you so desire.



When you provide a `geni-lib` script, you will see a slightly different set of buttons on the Create Profile page; next to the "Source" button there is an "XML" button that will pop up the RSpec XML for you to look at. The XML is read-only; if you want to change the profile, you will need to change the python source code that is displayed when you click on the "Source" button. Each time you change the python source code, the script is uploaded to the server and processed. Be sure to save your changes if you are updating an existing profile.

The following examples demonstrate basic `geni-lib` usage. More information about `geni-lib` and additional examples, can be found in the `geni-lib` repository. Its full documentation is online at geni-lib.readthedocs.org.

5.3.1 A single XEN VM node

```
# Import the Portal object.
import geni.portal as portal
# Import the ProtoGENI library.
import geni.rspec.pg as pg

# Create the Portal context.
pc = portal.Context()

# Create a Request object to start building the RSpec.
```

```
rspec = pg.Request()

# Create a XenVM and add it to the RSpec.
node = pg.XenVM("node")
rspec.addResource(node)

# Print the RSpec to the enclosing page.
pc.printRequestRSpec(rspec)
```

5.3.2 A single physical host

```
# Import the Portal object.
import geni.portal as portal
# Import the ProtoGENI library.
import geni.rspec.pg as pg

# Create the Portal context.
pc = portal.Context()

# Create a Request object to start building the RSpec.
rspec = pg.Request()

# Create a raw PC and add it to the RSpec.
node = pg.RawPC("node")
rspec.addResource(node)

# Print the RSpec to the enclosing page.
pc.printRequestRSpec(rspec)
```

5.3.3 Two XenVM nodes with a LAN between them

```
import geni.portal as portal
import geni.rspec.pg as pg

pc = portal.Context()
rspec = pg.Request()

# Create a XenVM nodes.
node1 = pg.XenVM("node1")
node2 = pg.XenVM("node2")

# Create interfaces for each node.
```

```
iface1 = node1.addInterface("if1")
iface2 = node2.addInterface("if2")

rspec.addResource(node1)
rspec.addResource(node2)

# Create a link with the type of LAN.
link = pg.LAN("lan")

# Add both node interfaces to the link.
link.addInterface(iface1)
link.addInterface(iface2)

# Add the link to the RSpec.
rspec.addResource(link)

pc.printRequestRSpec(rspec)
```

5.3.4 Two ARM64 servers in a LAN

```
import geni.portal as portal
import geni.rspec.pg as pg

pc = portal.Context()
rspec = pg.Request()

# Create two raw "PC" nodes
node1 = pg.RawPC("node1")
node2 = pg.RawPC("node2")

# Set each of the two to specifically request "m400" nodes, which
in CloudLab, are ARM
node1.hardware_type = "m400"
node2.hardware_type = "m400"

# Create interfaces for each node.
iface1 = node1.addInterface("if1")
iface2 = node2.addInterface("if2")

rspec.addResource(node1)
rspec.addResource(node2)

# Create a link with the type of LAN.
link = pg.LAN("lan")
```

```
# Add both node interfaces to the link.
link.addInterface(iface1)
link.addInterface(iface2)

# Add the link to the RSpec.
rspec.addResource(link)

pc.printRequestRSpec(rspec)
```

5.3.5 Set a specific IP address on each node

```
import geni.portal as portal
import geni.rspec.pg as pg

pc = portal.Context()
rspec = pg.Request()

node1 = pg.XenVM("node1")
iface1 = node1.addInterface("if1")

# Specify the component id and the IPv4 address
iface1.component_id = "eth1"
iface1.addAddress(pg.IPv4Address("192.168.1.1", "255.255.255.0"))

rspec.addResource(node1)

node2 = pg.XenVM("node2")
iface2 = node2.addInterface("if2")

# Specify the component id and the IPv4 address
iface2.component_id = "eth2"
iface2.addAddress(pg.IPv4Address("192.168.1.2", "255.255.255.0"))

rspec.addResource(node2)

link = pg.LAN("lan")

link.addInterface(iface1)
link.addInterface(iface2)

rspec.addResource(link)

pc.printRequestRSpec(rspec)
```

5.3.6 Specify an operating system and set install and execute scripts

```
import geni.portal as portal
import geni.rspec.pg as pg

pc = portal.Context()
rspec = pg.Request()

node1 = pg.XenVM("node1")

# Specify the URL for the disk image
node1.disk_image = "<URL to disk image>"

# Install and execute scripts on the VM
node1.addService(pg.Install(url="<URL to the tarball file>",
path="local"))
node1.addService(pg.Execute(shell="bash", command="<Path to exe-
cutable>"))

rspec.addResource(node1)
pc.printRequestRSpec(rspec)
```

5.3.7 A profile with parameters

```
import geni.portal as portal
import geni.rspec.pg as RSpec

# The legal types of nodes - note that the first string in each
pair is the
# string that will get put in the RSpec, and the second is the one
that will
# get shown to the user. If there is no need to show the user a
different string,
# each element of the array can be a simple string rather than a
tuple.
hw_types = [("m400", "ARM64"), ("dl360", "x86-64")]

pc = portal.Context()

# Define parameters - see this page for API documentation for the
# defineParameter() call:
#    http://geni-lib.readthedocs.org/en/latest/api/geniportal.html#geni.portal.Context.defin
pc.defineParameter("N", "Number of nodes",
                   portal.ParameterType.INTEGER, 5)
```

```
pc.defineParameter("hwtype", "Hardware type",
                   portal.ParameterType.NODETYPE, "m400",
hw_types)
pc.defineParameter("lan",  "Put all nodes in a LAN",
                   portal.ParameterType.BOOLEAN, False)

# Get values for the parameters - since a default is required by
defineParameter,
# the params object is guaranteed to contain a vale for every pa-
rameter
params = pc.bindParameters()

rspec = RSpec.Request()

# Create a LAN iff one was requested
if (params.lan):
    lan = RSpec.LAN()
    rspec.addResource(lan)

# Loop through the number of nodes requested, settting up each and
adding it
# to the RSpec
for i in range(1, params.N+1):
  node = RSpec.RawPC("node%d" % i)
  node.hardware_type = params.hwtype
  rspec.addResource(node)

  # Only create an interface if we are putting all nodes into a
LAN
  if params.lan:
    iface = node.addInterface("eth0")
    lan.addInterface(iface)

pc.printRequestRSpec(rspec)
```

5.4 Creating a profile from scratch

There are planned features relating to this section: see "§9.3 “Easier From-Scratch Profile Creation”" for more details.

CloudLab profiles are described by GENI RSpecs. You can create a profile directly from an RSpec by using the “Create Profile” option from the “Actions” menu. Note that you cannot edit the text fields until you upload an RSpec, as these fields edit (in-browser) fields in the RSpec.

5.5 Sharing Profiles

There are planned features relating to this section: see "§9.1 “Versioned Profiles”" for more details.

If you chose to make your profile publicly visible, it will show up in the main “Select Profile” list on `https://www.cloudlab.us/`. CloudLab also gives you direct links to your profiles so that you can share them with others, post them on your website, publish them in papers, etc. The link can be found on the profile’s detail page, which is linked for your “My Profiles” page. If you chose to make your profile accessible to anyone, the link will take the form `https://www.cloudlab.us//p/<project-id>/<profile-id>`. If you didn’t make the profile public, the URL will have the form `https://www.cloudlab.us//p/<UUID>`, where `UUID` is a 128-bit number so that the URL is not guessable. You can still share this URLs with anyone you want to have access to the profile—for example, to give it to a collaborator to try out your work before publishing.

6 Basic Concepts

This chapter covers the basic concepts that you'll need to understand in order to use CloudLab.

6.1 Profiles

A *profile* encapsulates everything needed to run an *experiment*. It consists of two main parts: a description of the resources (hardware, storage, network, etc.) needed to run the experiment, and the software artifacts that run on those resources.

The resource specification is in the RSpec format. The RSpec describes an entire *topology*: this includes the nodes (hosts) that the software will run on, the storage that they are attached to, and the network that connects them. The nodes may be virtual machines or physical servers. The RSpec can specify the properties of these nodes, such as how much RAM they should have, how many cores, etc., or can directly reference a specific class of hardware available in one of CloudLab's clusters. The network topology can include point to point links, LANs, etc. and may be either built from Ethernet or Infiniband.

The primary way that software is associated with a profile are through disk images. A disk image (often just called an "image") is a block-level snapshot of the contents of a real or virtual disk—and it can be loaded onto either. A disk image in CloudLab typically has an installation of a full operating system on it, plus additional packages, research software, data files, etc. that comprise the software environment of the profile. Each node in the RSpec is associated with a disk image, which it boots from; more than one node in a given profile may reference the same disk image, and more than one profile may use the same disk image as well.

Profiles come from two sources: some are provided by CloudLab itself; these tend to be standard installations of popular operating systems and software stacks. Profiles may also be provided by CloudLab's users, as a way for communities to share research artifacts.

6.1.1 On-demand Profiles

Profiles in CloudLab may be *on-demand profiles*, which means that they are designed to be instantiated for a relatively short period of time (hours or days). Each person instantiating the profile gets their own experiment, so everyone using the profile is doing so independently on their own set of resources.

6.1.2 Persistent Profiles

CloudLab also supports *persistent* profiles, which are longer-lived (weeks or months) and are set up to be shared by multiple users. A persistent profile can be thought of as a "testbed within a testbed"—a testbed facility built on top of CloudLab's hardware and provisioning capabilities. Examples of persistent profiles include:

- An instance of a cloud software stack, providing VMs to a large community
- A cluster set up with a specific software stack for a class
- A persistent instance of a database or other resource used by a large research community
- Machines set up for a contest, giving all participants access to the same hardware
- An HPC cluster temporarily brought up for the running of a particular set of jobs

A persistent profile may offer its own user interface, and its users may not necessarily be aware that they are using CloudLab. For example, a cloud-style profile might directly offer its own API for provisioning virtual machines. Or, an HPC-style persistent profile might run a standard cluster scheduler, which users interact with rather than the CloudLab website.

6.2 Experiments

See the chapter on repeatability for more information on repeatable experimentation in CloudLab.

An *experiment* is an instantiation of a profile. An experiment uses resources, virtual or physical, on one or more of the clusters that CloudLab has access to. In most cases, the resources used by an experiment are devoted to the individual use of the user who instantiates the experiment. This means that no one else has an account, access to the filesystems, etc. In the case of experiments using solely physical machines, this also means strong performance isolation from all other CloudLab users. (The exceptions to this rule are persistent profiles, which may offer resources to many users.)

Running experiments on CloudLab consume **real resources**, which are **limited**. We ask that you be careful about not holding on to experiments when you are not actively using them. If you are are holding on to experiments because getting your working environment set up takes time, consider creating a profile.

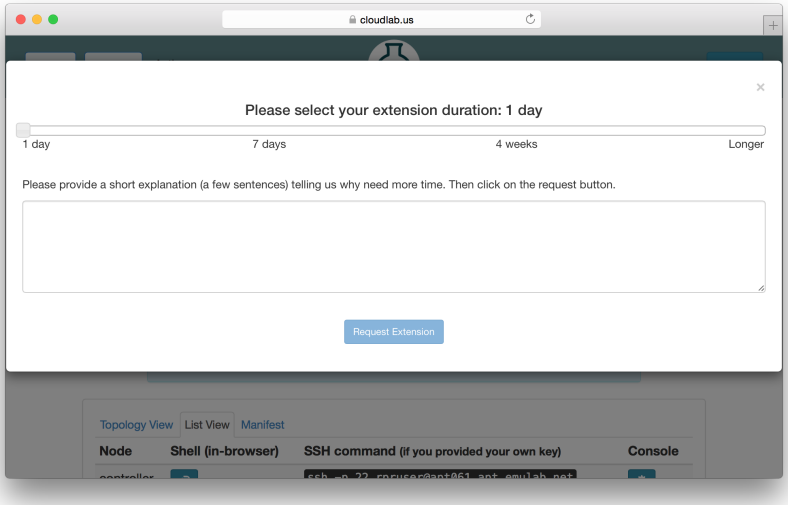
There are planned features relating to this section: see "§9.2 "Persistent Storage"" for more details.

The contents of local disk on nodes in an experiment are considered *ephemeral*—that is, when the experiment ends (either by being explicitly terminated by the user or expiring), the contents of the disk are lost. So, you should copy important data off before the experiment ends. A simple way to do this is through `scp` or `sftp`. You may also create a profile, which captures the contents of the disk in a disk image.

All experiments have an *expiration time*. By default, the expiration time is short (a few hours), but users can use the "Extend" button on the experiment page to request an extension. A request for an extension must be accompanied by a short description that explains the reason for requesting an extension, which will be reviewed by CloudLab staff. You will receive email a few hours before your experiment expires reminding you to copy your data off or request an extension.

6.2.1 Extending Experiments

If you need more time to run an experiment, you may use the "extend" button on the experiment's page. You will be presented with a dialog that allows you to select how much longer you need the experiment. Longer time periods require more extensive appoval processes. Short extensions are auto-approved, while longer ones require the intervention of CloudLab staff or, in the case of indefinite extensions, the steering commitee.



6.3 Projects

Users are grouped into *projects*. A project is, roughly speaking, a group of people working together on a common research or educational goal. This may be people in a particular research lab, a distributed set of collaborators, instructors and students in a class, etc.

A project is headed by a *project leader*. We require that project leaders be faculty, senior research staff, or others in an authoritative position. This is because we trust the project

leader to approve other members into the project, ultimately making them responsible for the conduct of the users they approve. If CloudLab staff have questions about a project's activities, its use of resources, etc., these questions will be directed to the project leader. Some project leaders run a lot of experiments themselves, while some choose to approve accounts for others in the project, who run most of the experiments. Either style works just fine in CloudLab.

Permissions for some operations / objects depend on the project that they belong to. Currently, the only such permission is the ability to make a profile visible onto to the owning project. We expect to introduce more project-specific permissions features in the future.

6.4 Virtual Machines

While CloudLab does have the ability to provision virtual machines itself (using the Xen hypervisor), we expect that the dominant use of CloudLab is that users will provision physical machines. Users (or the cloud software stacks that they run) may build their own virtual machines on these physical nodes using whatever hypervisor they wish.

6.5 Physical Machines

Users of CloudLab may get exclusive, root-level control over *physical machines*. When allocated this way, no layers of virtualization or indirection get in the way of the way of performance, and users can be sure that no other users have access to the machines at the same time. This is an ideal situation for repeatable research.

Physical machines are re-imaged between users, so you can be sure that your physical machines don't have any state left around from the previous user. You can find descriptions of the hardware in CloudLab's clusters in the hardware chapter.

7 Advanced Topics

This section is under construction

7.1 Disk Images

Disk images in CloudLab are stored and distributed in the Frisbee disk image format. They are stored at block level, meaning that, in theory, any filesystem can be used. In practice, Frisbee's filesystem-aware compression is used, which causes the image snapshotting and installation processes to parse the filesystem and skip free blocks; this provides large performance benefits and space savings. Frisbee has support for filesystems that are variants of the BSD UFS/FFS, Linux ext, and Windows NTFS formats. The disk images created by Frisbee are bit-for-bit identical with the original image, with the caveat that free blocks are skipped and may contain leftover data from previous users.

Disk images in CloudLab are typically created by starting with one of CloudLab's supplied images, customizing the contents, and taking a snapshot of the resulting disk. The snapshotting process reboots the node, as it boots into an MFS to ensure a quiescent disk. If you wish to bring in an image from outside of CloudLab or create a new one from scratch, please contact us for help; if this is a common request, we may add features to make it easier.

CloudLab has default disk image for each node type; after a node is freed by one experimenter, it is re-loaded with the default image before being released back into the free pool. As a result, profiles that use the default disk images typically instantiate faster than those that use custom images, as no disk loading occurs.

Frisbee loads disk images using a custom multicast protocol, so loading large numbers of nodes typically does not slow down the instantiation process much.

Images may be referred to in requests in three ways: by URN, by an unqualified name, and by URL. URNs refer to a specific image that may be hosted on any of the CloudLab-affilated clusters. An unqualified name refers to the version of the image hosted on the cluster on which the experiment is instantiated. If you have large images that CloudLab cannot store due to space constraints, you may host them yourself on a webserver and put the URL for the image into the profile. CloudLab will fetch your image on demand, and cache it for some period of time for efficient distribution.

Images in CloudLab are versioned, and CloudLab records the provenance of images. That is, when you create an image by snapshotting a disk that was previously loaded with another image, CloudLab records which image was used as a base for the new image, and stores only the blocks that differ between the two. Image URLs and URNs can contain version numbers, in which case they refer to that specific version of the image, or they may omit the version number, in which case they refer to the latest version of the image at the time an experiment is instantiated.

7.2 RSpecs

The resources (nodes, links, etc.) that define a profile are expressed in the RSpec format from the GENI project. In general, RSpec should be thought of as a sort of "assembly language"—something it's best not to edit yourself, but as manipulate with other tools or create as a "compiled" target from a higher-level language.

That said, the tools for manipulating RSpecs are still incomplete. (For a preview of CloudLab's plans in this regard, see our section on planned profile creation features.) So, there are still some cases in which it is unfortunately useful to look at or manipulate RSpecs directly.

This section is under construction *Still to come: documentation of CloudLab's extensions to the RSpec format.*

7.3 Public IP Access

There are planned features relating to this section: see "§9.10 "Dynamic Public IP Addresses"" for more details.

CloudLab treats publicly-routable IP addresses as an allocatable resource.

By default, all physical hosts are given a public IP address. This IP address is determined by the host, rather than the experiment. There are two DNS names that point to this public address: a static one that is the node's permanent hostname (such as `apt042.apt.emulab.net`), and a dynamic one that is assigned based on the experiment; this one may look like `<vname>.<exp>.<proj>.apt.emulab.net`, where `<vname>` is the name assigned in the request RSpec, `<eid>` is the name assigned to the experiment, and `proj` is the project that the experiment belongs to. This name is predictable regardless of the physical nodes assigned.

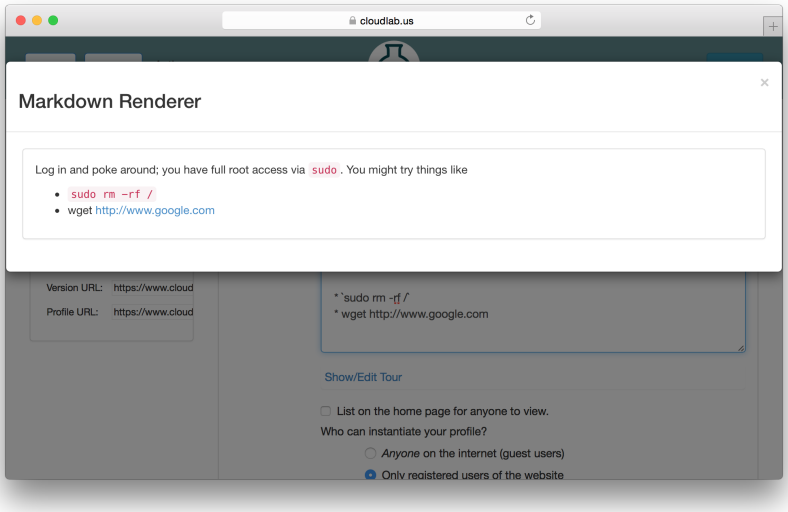
By default, virtual machines are *not* given public IP addresses; basic remote access is provided through an `ssh` server running on a non-standard port, using the physical host's IP address. This port can be discovered through the manifest of an instantiated experiment, or on the "list view" of the experiment page. If a public IP address is required for a virtual machine (for example, to host a webserver on it), a public address can be requested on a per-VM basis. If using the Jacks GUI, each VM has a checkbox to request a public address. If using python scripts and `geni-lib`, setting the `routable_control_ip` property of a node accomplishes the same effect. Different clusters will have different numbers of public addresses available for allocation in this manner.

Note that the above refers to VMs created by CloudLab itself; for CloudLab users who wish to bring up their own virtual machines and wish to have public addresses assigned to them, see our section on planned support for dynamic public addresses.

7.4 Markdown

CloudLab supports Markdown in the major text fields in RSpecs. Markdown is a simple formatting syntax with a straightforward translation to basic HTML elements such as headers, lists, and pre-formatted text. You may find this useful in the description and instructions attached to your profile.

While editing a profile, you can preview the Markdown rendering of the Instructions or Description field by double-clicking within the text box.



You will probably find the Markdown manual to be useful.

7.5 Tours

This feature under development

8 Hardware

CloudLab can allocate experiments on any one of several federated clusters.

CloudLab is in the early phases of construction; please see `https://www.cloudlab.us/#availability` **for an up to date deployment schedule.**

CloudLab has the ability to dispatch experiments to several clusters. Additional hardware expansions are planned, and descriptions of them can be found at `https://www.cloudlab.us/hardware.php`

8.1 Utah/HP CloudLab Cluster

The CloudLab cluster at the University of Utah is being built in partnership with HP. The first phase of this cluster consists of 315 64-bit ARM servers with 8 cores each, for a total of 2,520 cores. The servers are built on HP's Moonshot platform using X-GENE system-on-chip designs from Applied Micro. The cluster is hosted in the University of Utah's Downtown Data Center in Salt Lake City.

More technical details can be found at `https://www.cloudlab.us/hardware.php#utah`

m400	315 nodes
CPU	Eight 64-bit ARMv8 (Atlas/A57) cores at 2.4 GHz (APM X-GENE)
RAM	64GB ECC Memory (8x 8 GB DDR3-1600 SO-DIMMs)
Disk	120 GB of flash (SATA3 / M.2, Micron M500)
NIC	Dual-port Mellanox ConnectX-3 10 GB NIC (PCIe v3.0, 8 lanes

There are 45 nodes in a chassis, and this cluster consists of seven chassis. Each chassis has two 45XGc switches; each node is connected to both switches, and each chassis switch has four 40Gbps uplinks, for a total of 320Gbps of uplink capacity from each chassis. One switch is used for control traffic, connecting to the Internet, etc. The other is used to build experiment topologies, and should be used for most experimental purposes.

All chassis are interconnected through a large HP FlexFabric 12910 switch which has full bisection bandwidth internally.

We have plans to enable some users to allocate entire chassis; when allocated in this mode, it will be possible to have complete administrator control over the switches in addition to the nodes.

8.2 Wisconsin/Cisco CloudLab Cluster

This cluster is available in "alpha" status: auto-configuration of the network is not yet supported, but is expected soon.

The CloudLab cluster at the University of Wisconsin is being built in partnership with Cisco and Seagate. The initial cluster will have 100 servers with a total of 1,600 cores connected in a CLOS topology with full bisection bandwidth. It will have 525 TB of storage, including SSDs on every node.

More technical details can be found at `https://www.cloudlab.us/hardware.php#wisconsin`

ucs220m4	90 nodes
CPU	Two Intel E5-2630 v3 8-core CPUs at 2.40 GHz (Haswell w/ EM64T)
RAM	128GB ECC Memory (8x 16 GB DDR4 2133 MHz PC4-17000 dual rank RDIMMs
Disk	Two 1.2 TB 10K RPM 6G SAS SFF HDDs
Disk	One 480 GB 6G SAS SSD
NIC	Dual-port Cisco VIC1227 10Gb NIC (PCIe v3.0, 8 lanes
NIC	Onboard Intel i350 1Gb

ucs240m4	10 nodes
CPU	Two Intel E5-2630 v3 8-core CPUs at 2.40 GHz (Haswell w/ EM64T)
RAM	128GB ECC Memory (8x 16 GB DDR4 2133 MHz PC4-17000 dual rank RDIMMs
Disk	One 1 TB 7.2K RPM SAS 3.5" HDD
Disk	One 480 GB 6G SAS SSD
Disk	Twelve 3 TB 3.5" HDDs donated by Seagate
NIC	Dual-port Cisco VIC1227 10Gb NIC (PCIe v3.0, 8 lanes
NIC	Onboard Intel i350 1Gb

All nodes are connected to two networks:

- A 1 Gbps Ethernet **"control network"**—this network is used for remote access, experiment management, etc., and is connected to the public Internet. When you log in to nodes in your experiment using `ssh`, this is the network you are using. *You should not use this network as part of the experiments you run in CloudLab.*
- A 10 Gbps Ethernet **"experiment network"**–each node has **two interfaces** on this network. Twelve leaf switches are Cisco Nexus C3172PQs, which have 48 10Gbps ports for the nodes and six 40Gbps uplink ports. They are connected to six spine switches (Cisco Nexus C3132Qs); each leaf has one 40Gbps link to each spine switch. Another C3132Q switch acts as a core; each spine switch has one 40Gbps link to it, and it has upstream links to Internet2.

8.3 Clemson/Dell CloudLab Cluster

This cluster is on-site in Clemson, and is in the process of being set up.

The CloudLab cluster at Clemson University is being built in partnership with Dell. The initial cluster will have 100 servers with a total of 2,000 cores, 424TB of disk space, and

26TB of RAM. All nodes have both Ethernet and Infiniband networks.

More technical details can be found at `https://www.cloudlab.us/hardware.php#clemson`

c8220	96 nodes
CPU	Two Intel E5-2660 v2 10-core CPUs at 2.20 GHz (Ivy Bridge)
RAM	256GB ECC Memory (16x 16 GB DDR4 1600MT/s dual rank RDIMMs
Disk	Two 1 TB 7.2K RPM 3G SATA HDDs
NIC	Dual-port Intel 10Gbe NIC (PCIe v3.0, 8 lanes
NIC	Qlogic QLE 7340 40 Gb/s Infiniband HCA (PCIe v3.0, 8 lanes)

c8220x	4 nodes
CPU	Two Intel E5-2660 v2 10-core CPUs at 2.20 GHz (Ivy Bridge)
RAM	256GB ECC Memory (16x 16 GB DDR4 1600MT/s dual rank RDIMMs
Disk	Eight 1 TB 7.2K RPM 3G SATA HDDs
Disk	Twelve 4 TB 7.2K RPM 3G SATA HDDs
NIC	Dual-port Intel 10Gbe NIC (PCIe v3.0, 8 lanes
NIC	Qlogic QLE 7340 40 Gb/s Infiniband HCA (PCIe v3.0, 8 lanes)

All nodes are connected to three networks:

- A 1 Gbps Ethernet **"control network"**—this network is used for remote access, experiment management, etc., and is connected to the public Internet. When you log in to nodes in your experiment using `ssh`, this is the network you are using. *You should not use this network as part of the experiments you run in CloudLab.*
- A 10 Gbps Ethernet **"experiment network"**–each node has **one interface** on this network. Three Dell Force10 S6000 switches are used to implent it: two are used to connect nodes directly, while the third is used as an aggregator connecting the two leaf switches. Each S6000 has 32 40Gbps ports. On the leaf switches, these are broken out as 96 10Gbps ports, plus 8 40 Gbps uplink ports. This gives a 3:1 blocking factor between the two leaf switches.
- A 40 Gbps QDR Infiniband **"experiment network"**–each has one connection to this network, which is implemented using a large Mellanox chassis switch with full bisection bandwidth.

8.4 Apt Cluster

The main Apt cluster is housed in the University of Utah's Downtown Data Center in Salt Lake City, Utah. It contains two classes of nodes:

r320	128 nodes
CPU	1x Xeon E5-2450 processor (8 cores, 2.1Ghz)
RAM	16GB Memory (4 x 2GB RDIMMs, 1.6Ghz)

Disks	4 x 500GB 7.2K SATA Drives (RAID5)
NIC	1GbE Dual port embedded NIC (Broadcom)
NIC	1 x Mellanox MX354A Dual port FDR CX3 adapter w/1 x QSA adapter

c6220	64 nodes
CPU	2 x Xeon E5-2650v2 processors (8 cores each, 2.6Ghz)
RAM	64GB Memory (8 x 8GB DDR-3 RDIMMs, 1.86Ghz)
Disks	2 x 1TB SATA 3.5" 7.2K rpm hard drives
NIC	4 x 1GbE embedded Ethernet Ports (Broadcom)
NIC	1 x Intel X520 PCIe Dual port 10Gb Ethernet NIC
NIC	1 x Mellanox FDR CX3 Single port mezz card

All nodes are connected to three networks with **one interface each**:

- A 1 Gbps *Ethernet* **"control network"**—this network is used for remote access, experiment management, etc., and is connected to the public Internet. When you log in to nodes in your experiment using `ssh`, this is the network you are using. *You should not use this network as part of the experiments you run in Apt.*
- A **"flexible fabric"** that can run up to 56 Gbps and runs *either FDR Infiniband or Ethernet*. This fabric uses NICs and switches with Mellanox's VPI technology. This means that we can, on demand, configure each port to be either FDR Inifiniband or 40 Gbps (or even non-standard 56 Gbps) Ethernet. This fabric consists of seven edge switches (Mellanox SX6036G) with 28 connected nodes each. There are two core switches (also SX6036G), and each edge switch connects to both cores with a 3.5:1 blocking factor. This fabric is ideal if you need **very low latency, Infiniband, or a few, high-bandwidth Ethernet links**.
- A 10 Gbps *Ethernet* **"commodity fabric"**. One the `r320` nodes, a port on the Mellanox NIC (permanently set to Ethernet mode) is used to connect to this fabric; on the `c6220` nodes, a dedicated Intel 10 Gbps NIC is used. This fabric is built from two Dell Z9000 switches, each of which has 96 nodes connected to it. It is idea for creating **large LANs**: each of the two switches has full bisection bandwidth for its 96 ports, and there is a 3.5:1 blocking factor between the two switches.

8.5 IG-DDC Cluster

This small cluster is an InstaGENI Rack housed in the University of Utah's Downtown Data Center. It has nodes of only a single type:

dl360	33 nodes
CPU	2x Xeon E5-2450 processors (8 cores each, 2.1Ghz)
RAM	48GB Memory (6 x 8GB RDIMMs, 1.6Ghz)
Disk	1 x 1TB 7.2K SATA Drive
NIC	1GbE 4-port embedded NIC

It has two network fabrics:

- A 1 Gbps **"control network"**. This is used for remote access, and should not be used for experiments.
- A 1 Gbps **"experiment network"**. Each node has *three* interfaces on this network, which is built from a single HP Procurve 5406 switch. OpenFlow is available on this network.

9 Planned Features

This chapter describes features that are planned for CloudLab or under development: please contact us if you have any feedback or suggestions!

9.1 Versioned Profiles

We plan to add the *versioning* to profiles to capture the evolution of a profile over time. When updating profiles, the result will be a new version that does not (entirely) replace the profile being updated.

There will be two types of versions: *working* versions that should be considered ephemeral, and *published* versions that are intended to be long-term stable. For example, a user may generate many working versions as they refine their software, fix bugs, etc. Then, when the profile is in a state where it is appropriate to share with others, it can be published. Users will be able to link to a specific version—example, to unambiguously identify which version was used for a paper.

One limitation on this feature will be the fact that CloudLab has limited storage space; we will have to apply a quota system that limits the amount of storage that a project can use to store multiple versions of the same profile.

9.2 Persistent Storage

For the time being, the contents of all disks in CloudLab are considered ephemeral: the contents are lost whenever an experiment terminates. The only way to save data is to copy it off or to create a profile using the disk.

We plan to change this by adding persistent storage that is hosted on storage servers within CloudLab. Users will be able to use the CloudLab web interface to create and manage their persistent storage, and profiles will be able to reference where these stores should be mounted in the experiment. When sharing profiles, it will be possible to indicate that the persistent store may only be mounted read-only—a common use case will be to put a dataset in a persistent store, and allow other CloudLab users read-only access to the dataset.

There will be two types of persistent storage: *block stores* which are mounted using iSCSI, and which generally can only be mounted on one host at a time, and *file stores* which are mounted over NFS, and can be mounted read/write by many nodes at the same time.

This feature will be based on Emulab's block storage system. Underlying this system is ZFS, which supports snapshots. We intend to expose this snapshot functionality to users, and to allow profiles to mount specific snapshots (eg. the version of a dataset used for a particular paper.)

It should be noted that performance of persistent stores will not be guaranteed or isolated from other users, since it will be implemented using shared storage servers that others may be accessing at the same time. Therefore, for experiments whose repeatability depends on I/O performance, all data should be copied to local disk before use.

9.3 Easier From-Scratch Profile Creation

Currently, there are two ways to create profiles in CloudLab: cloning an existing profile or creating one from scratch by writing an RSpec by hand. We plan to add two more: a GUI for RSpec creation, and bindings to a programming language for generation of RSpecs.

The GUI will be based on Jacks, an embeddable RSpec editor currently in development for the GENI project. Jacks is already used in CloudLab to display topologies in the profile selection dialog and on the experiment page.

The programming language bindings will allow users to write programs in an existing, popular language (likely Python) to create RSpecs. This will allow users to use loops, conditionals, and other programming language constructs to create large, complicated RSpecs. We are evaluating `geni-lib` for this purpose.

9.4 "Quick" Profiles

Sometimes, you just need one node running a particular disk image, without making a complicated profile to go with it. We plan to add a "quick profile" feature that will create a one-off experiment with a single node.

9.5 Improved Physical Resource Descriptions

As part of the process of reserving resources on CloudLab, a type of RSpec called a manifest is created. The manifest gives a detailed description of the hardware allocated, including the specifics of network topology. Currently, CloudLab does not directly export this information to the user. In the interest of improving transparency and repeatable research, CloudLab will develop interfaces to expose, explore, and export this information.

9.6 Bare-metal Access to Switches

Today, switches in CloudLab are treated as infrastructure; that is, they are under CloudLab's control and, while we provide a high degree of transparency, we do not let users control them directly. We plan to make at least some switches—in most cases, ToRs, and in others, spine and/or core—directly accessible to users. This means that users will have console access to

the switches, and will be able to reconfigure them and/or load different versions of firmware on them. (This will only be offered on switches that are only used by a single experiment at a time.)

9.7 OpenFlow Support

All switches in CloudLab will be OpenFlow-capable. In the case of exclusive-access bare metal switches, users will get direct and complete OpenFlow access to the switches. In the case of shared switches, we are investigating the use of FlowSpace Firewall from the GRNOC and Internet2 for virtualization.

9.8 Switch Monitoring

We plan to export many of the monitoring features available in CloudLab's infrastructure switches—port counters, queue lengths, etc.

9.9 Power Monitoring

Some of the equipment in CloudLab will have the ability to take fine-grained measurements of power usage and other environmental sensors (such as temperature). CloudLab will provide both logged and real-time access to this data for experimenters.

9.10 Dynamic Public IP Addresses

In some cases, users would like to create their own virtual machines, and would like to give them public IP addresses. We plan to allow profiles to request a pool of dynamic addresses; VMs brought up by the user can then run DHCP to be assigned one of these addresses. An open question is how we will be able to tell which DHCP requests belong to which experiment.

10 CloudLab OpenStack Tutorial

This tutorial will walk you through the process of creating a small cloud on CloudLab using OpenStack. Your copy of OpenStack will run on bare-metal machines that are dedicated for your use for the duration of your experiment. You will have complete administrative access to these machines, meaning that you have full ability to customize and/or configure your installation of OpenStack.

10.1 Objectives

In the process of taking this tutorial, you will learn to:

- Log in to CloudLab using a GENI portal account
- Create your own cloud by using a pre-defined profile
- Access resources in a cloud that you create
- Use administrative access to customize your cloud
- Clean up your cloud when finished
- Learn where to get more information

10.2 Prerequisites

This tutorial assumes that:

- You have an existing account on the GENI portal. (Instructions for getting an account can be found here.)
- You have set up an `ssh` keypair in your GENI portal account. (When logged into the GENI portal, you can add a public key under "Profile -> SSH Keys".)
- You have an `ssh` client set up to use that keypair.

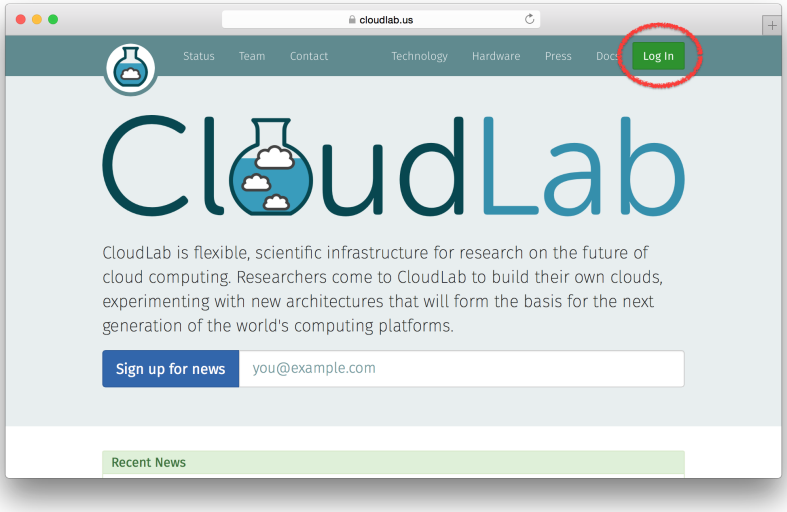
10.3 Logging In

The first step is to log in to CloudLab; CloudLab is available to all researchers and educators who work in cloud computing. If you have an account at one of its federated facilities, like GENI, then you already have an account at CloudLab.

This document assumes that you will log in using an existing GENI account. If you want to use a different account, or need to apply for one, see the chapter about user accounts.

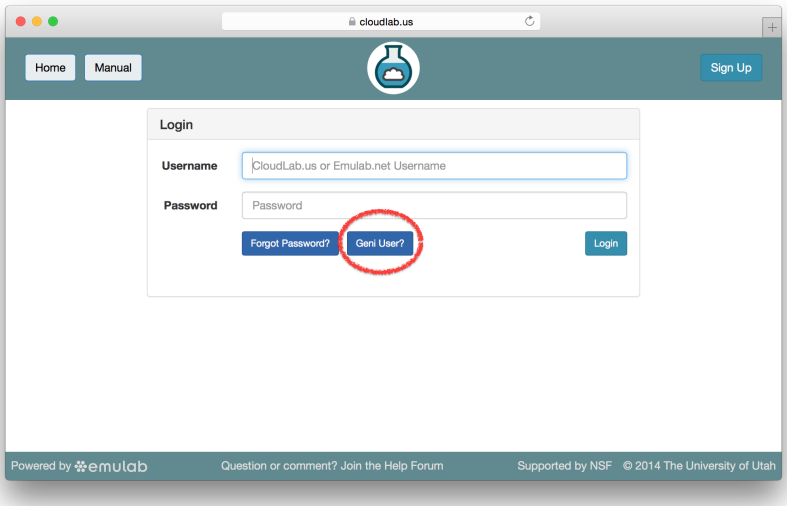
1. **Open the CloudLab web interface**

   To log in to CloudLab using a GENI account, start by visiting `https://www.cloudlab.us/` in your browser and using the "Log In" button in the upper right.



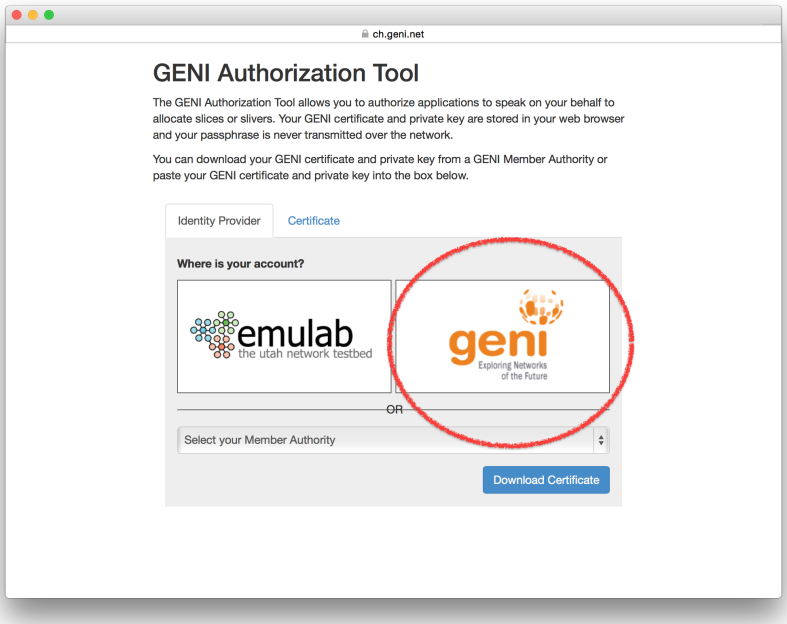
2. **Click the "GENI User" button**

   On the login page that appears, you will use the "GENI User" button. This will start the GENI authorization tool, which lets you connect your GENI account with CloudLab.



3. **Select the GENI portal**

   You will be asked which facility your account is with. Select the GENI icon, which will take you to the GENI portal. There are several other facilities that are federated with CloudLab, which can be found in the drop-down list.

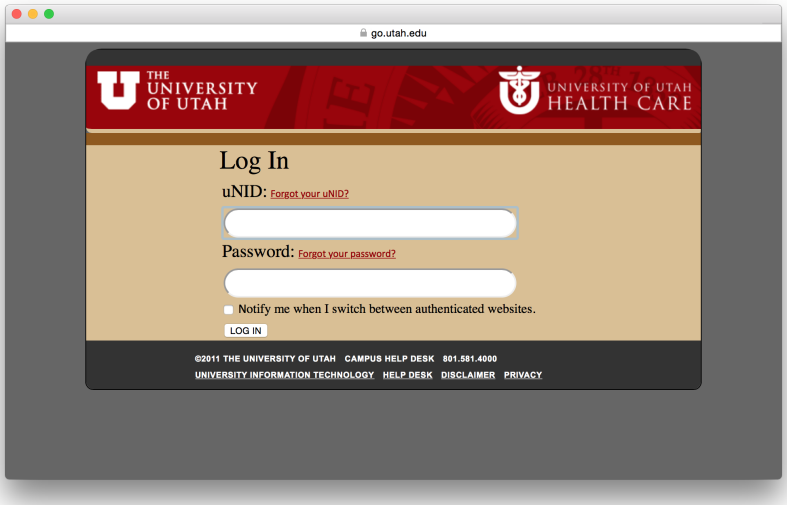


4. **Log into the GENI portal**

   You will be taken to the GENI portal, and will need to select the institution that is your identity provider. Usually, this is the university you are affiliated with. If your university is not in the list, you may need to log in using the "GENI Project Office" identity provider. If you have ever logged in to the GENI portal before, your identity provider should be pre-selected; if you are not familiar with this login page, there is a good chance that you don't have a GENI account and need to apply for one.

5. **Log into your institution**

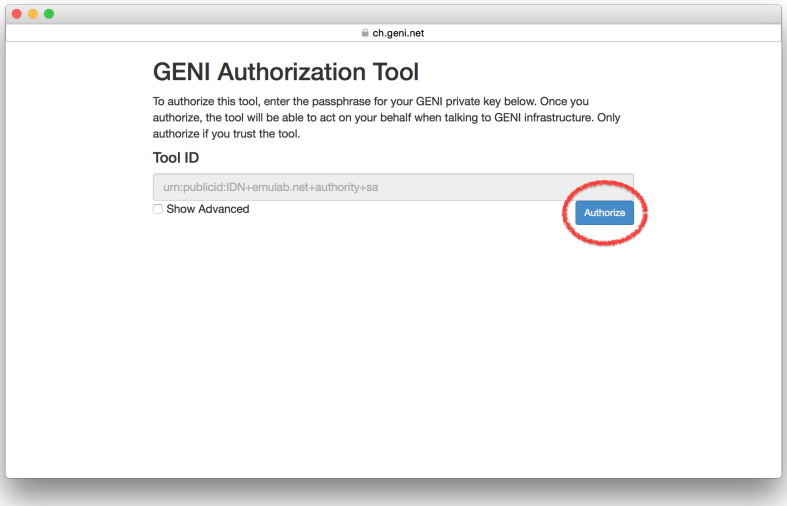
   This will bring you to your institution's standard login page, which you likely use to access many on-campus services. (Yours will look different from this one.)



6. **Authorize CloudLab to use your GENI account**

What's happening in this step is that your browser uses your GENI user certificate (which it obtained from the GENI portal) to cryptographically sign a "speaks-for" credential that authorizes the CloudLab portal to make GENI API calls on your behalf

Click the "Authorize" button: this will create a signed statement authorizing the CloudLab portal to speak on your behalf. This authorization is time-limited (see "Show Advanced" for the details), and all actions the CloudLab portal takes on your behalf are clearly traceable.



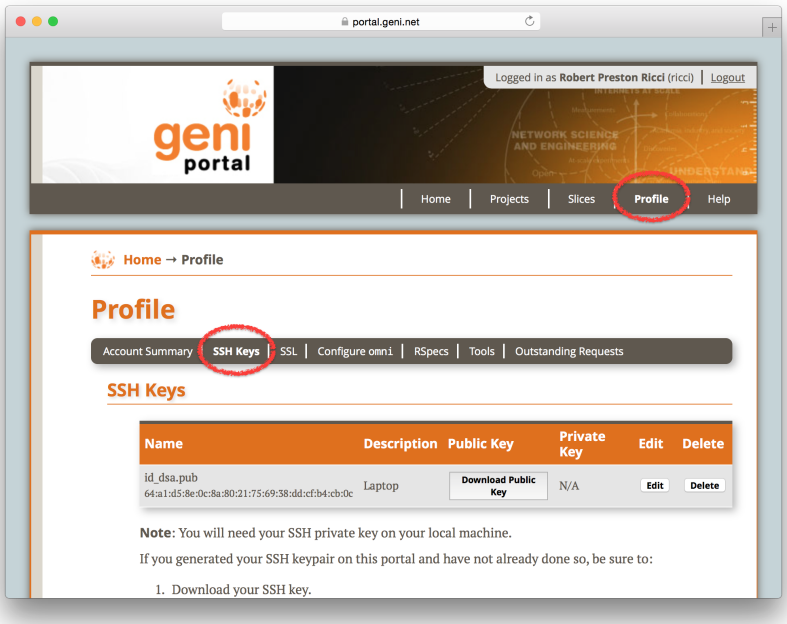
Note: We have heard from some Mac users who have applied Apple Security Update 2015-002 (March 9, 2015) that this step hangs, but that re-doing the login process a second time works fine.

Note: If you are unable to log in even after going through the process a second time, you can apply for a "native" CloudLab account by following this link. You may need to wait a few minutes for your account to be approved. Once it is, you will enter this new username and password directly into the CloudLab login form, rather than using the "GENI User" button.

7. **Set up an SSH keypair in the GENI portal**

   Though not strictly required, many parts of this tutorial will work better if you have an `ssh` public key associated with your GENI Portal account. To upload one, visit portal.geni.net and to the "Profile" page, "SSH Keys" tab.

   If you are not familiar with `ssh` keys, you may skip this step. You may find GitHub's `ssh` key page helpful in learning how to generate and use them.



10.4 Building Your Own OpenStack Cloud

Once you have logged in, you will "instantiate" a §6.1 "Profiles""profile" to create an experiment. (An experiment in CloudLab is similar to a "slice" in GENI.) Profiles are CloudLab's way of packaging up configurations and experiments so that they can be shared with others. Each experiment is separate: the experiment that you create for this tutorial will be an instance of a profile provided by the facility, but running on resources that are dedicated to you, which you have complete control over. This profile uses local disk space on the nodes, so anything you store there will be lost when the experiment terminates.

The OpenStack cloud we will build in this tutorial is very small, but CloudLab has large clusters that can be used for larger-scale experiments.

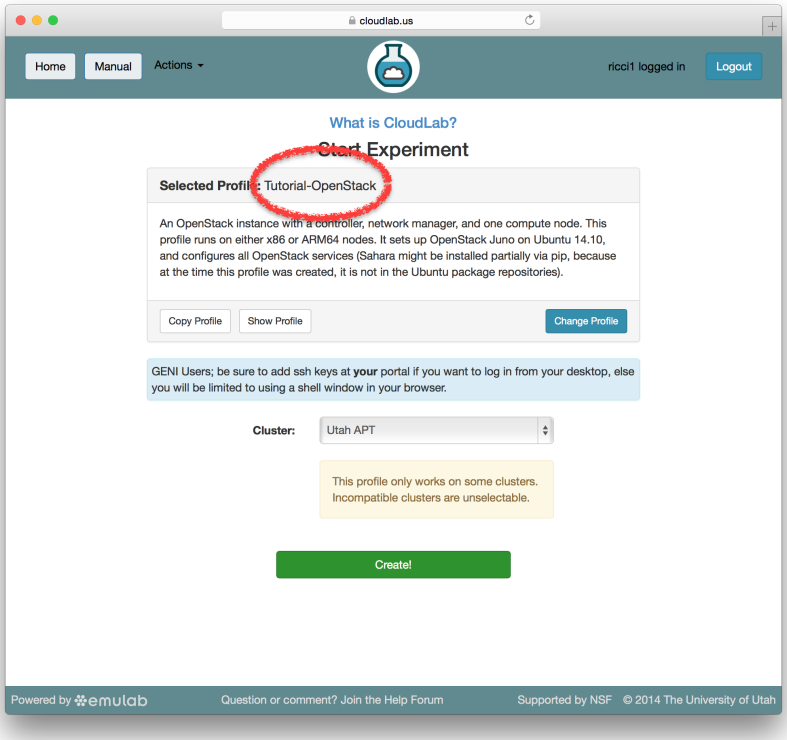
For this tutorial, we will use a basic profile that brings up a small OpenStack cloud. The CloudLab staff have built this profile by capturing disk images of a partially-completed OpenStack installation and scripting the remainder of the install (customizing it for the specific machines that will get allocated, the user that created it, etc.) See this manual's section on profiles for more information about how they work.

1. **Select a profile**

   The "Start an Experiment" page is where you will select a profile to instantiate. We will use the **Tutorial-OpenStack** profile; if it is not selected, follow this link or click

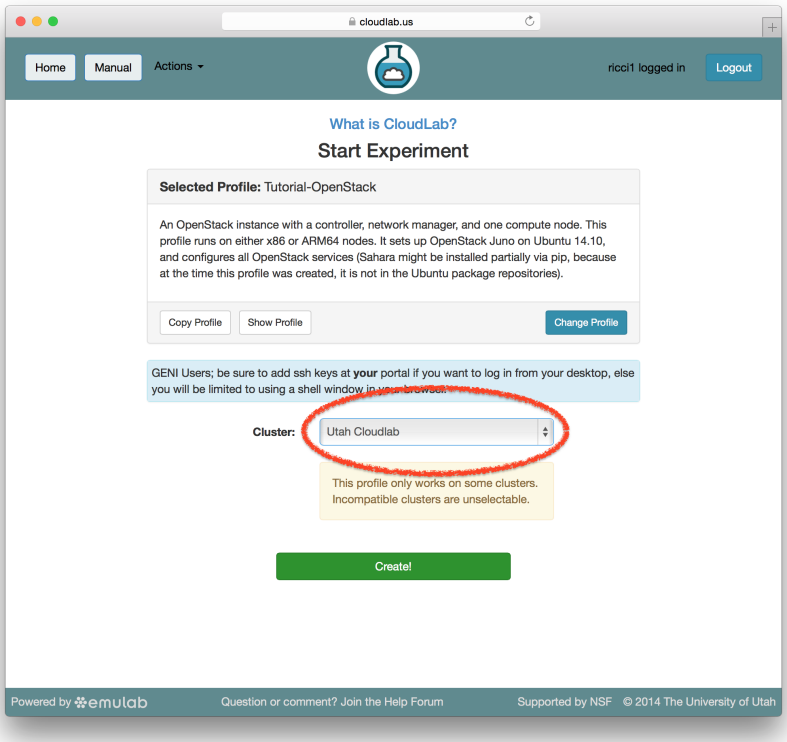
After logging in, you will be taken to the "Start an Experiment" page automatically if you do not have any current, running experiments. You can get back to this page at any time by selecting the "Start Experiment" link from the "Actions" menu.

the “Change Profile” button, and select “Tutorial-OpenStack” from the list on the left.



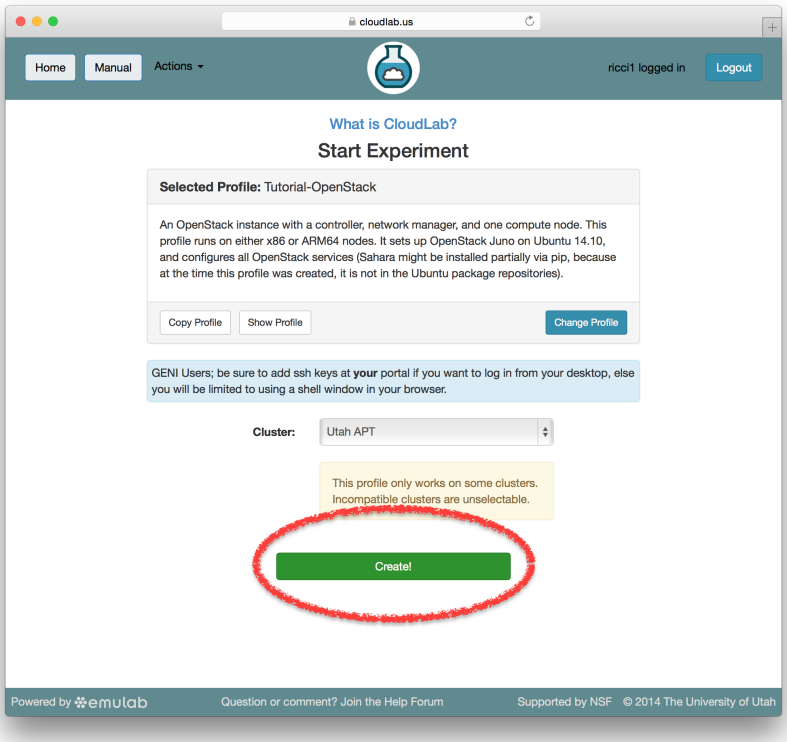
2. **Select a cluster**

   CloudLab has multiple clusters available to it. Some profiles can run on any cluster, some can only run on specific ones due to specific hardware constraints, etc. Make sure that the Utah CloudLab cluster is selected.



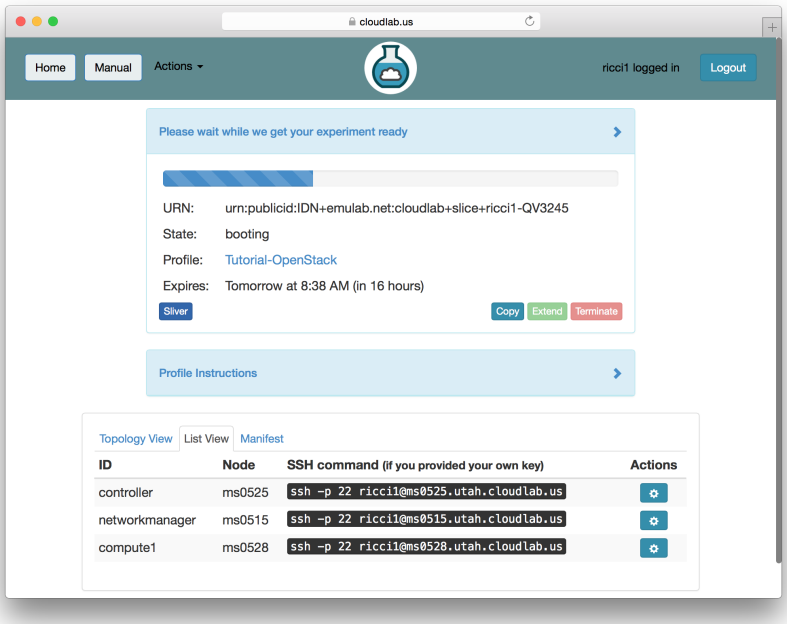
3. **Click Create!**

   When you click the "create" button, CloudLab will start provisioning the resources that you requested on the cluster that you selected.



4. **CloudLab instantiates your profile**

   CloudLab will take a few minutes to bring up your copy of OpenStack, as many things happen at this stage, including selecting suitable hardware, loading disk images on local storage, booting bare-metal machines, re-configuring the network topology, etc. While this is happening, you will see this status page:



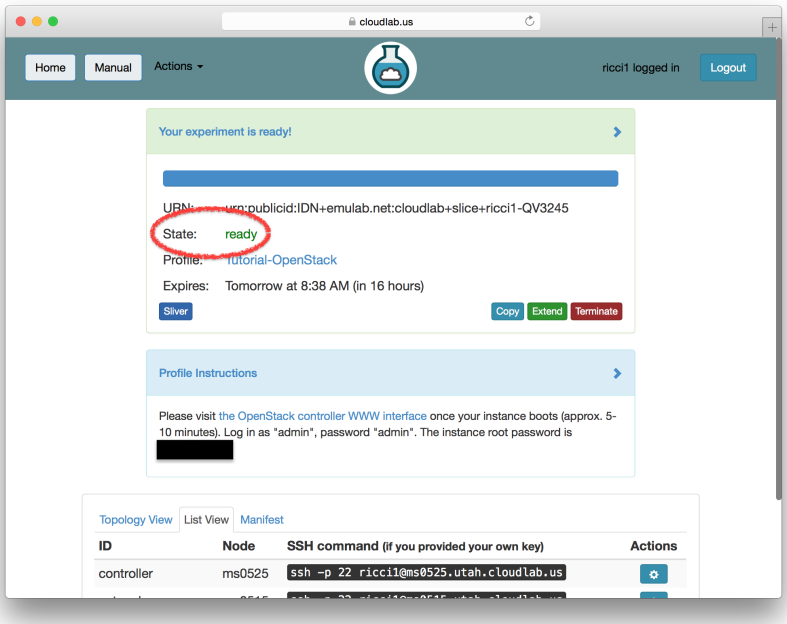
As soon as a set of resources have been assigned to you, you will see details about them at the bottom of the page (though you will not be able to log in until they have gone through the process of imaging and booting.) While you are waiting for your resources to become available, you may want to have a look at the CloudLab user manual, or use the "Sliver" button to watch the logs of the resources ("slivers") being provisioned and booting.

Provisioning is done using the GENI APIs; it is possible for advanced users to bypass the CloudLab portal and call these provisioning APIs from their own code.

5. **Your cloud is ready!**

When the web interface reports the state as "Ready", your cloud is provisioned, and you can proceed to the next section.

Important: A "Ready" status indicates that resources are provisioned and booted; this particular profile runs scripts to complete the OpenStack setup, and it will be a few more minutes before OpenStack is fully ready to log in and create virtual machine instances. For now, don't attempt to log in to OpenStack, we will explore the CloudLab experiment first.

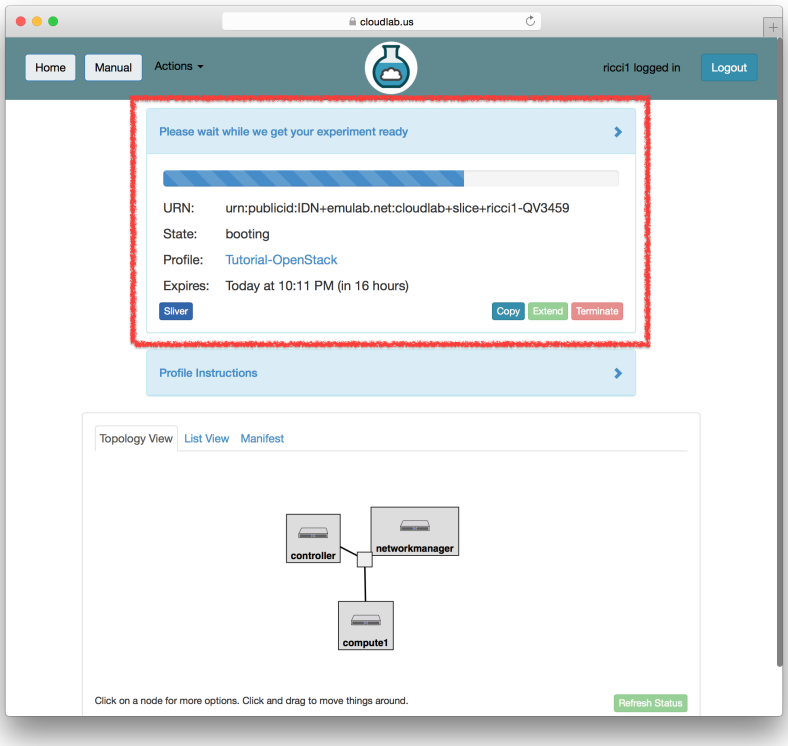


10.5 Exploring Your Experiment

Now that your experiment is ready, take a few minutes to look at various parts of the CloudLab status page to help you understand what resources you've got and what you can do with them.

10.5.1 Experiment Status

The panel at the top of the page shows the status of your experiment—you can see which profile it was launched with, when it will expire, etc. The buttons in this area let you make a copy of the profile (so that you can customize it), ask to hold on to the resources for longer, or release them immediately.

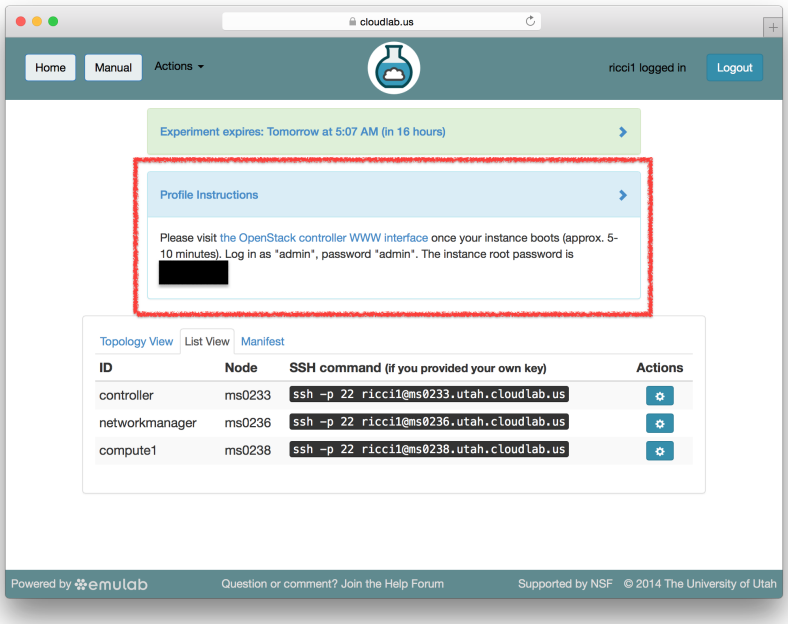


Note that the default lifetime for experiments on CloudLab is less than a day; after this time, the resources will be reclaimed and their disk contents will be lost. If you need to use them for longer, you can use the "Extend" button and provide a description of why they are needed. Longer extensions require higher levels of approval from CloudLab staff. You might also consider creating a profile of your own if you might need to run a customized environment multiple times or want to share it with others.

You can click the title of the panel to expand or collapse it.

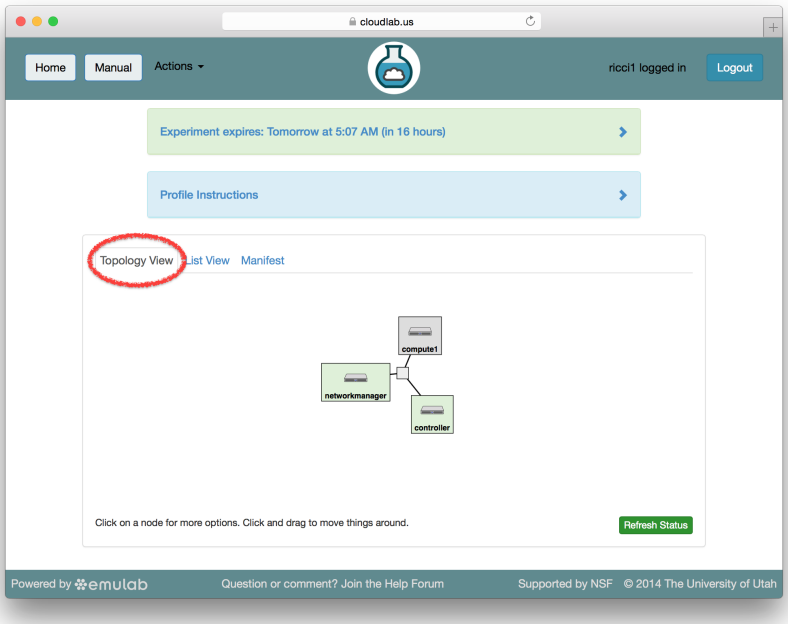
10.5.2 Profile Instructions

Profiles may contain written instructions for their use. Clicking on the title of the "Profile Instructions" panel will expand (or collapse) it; in this case, the instructions provide a link to the administrative interface of OpenStack, and give you passwords to use to log in. (Don't log into OpenStack yet—for now, let's keep exploring the CloudLab interface.)



10.5.3 Topology View

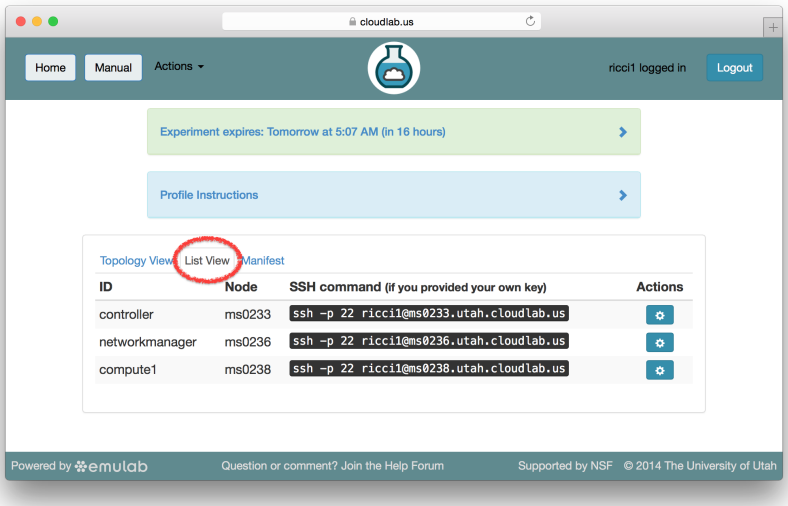
At the bottom of the page, you can see the topology of your experiment. This profile has three nodes connected by a 10 Gigabit LAN, which is represented by a gray box in the middle of the topology. The names given for each node are the names assigned as part of the profile; this way, every time you instantiate a profile, you can refer to the nodes using the same names, regardless of which physical hardware was assigned to them. The green boxes around each node indicate that they are up; click the "Refresh Status" button to initiate a fresh check.



It is important to note that every node in CloudLab has at least *two* network interfaces: one "control network" that carries public IP connectivity, and one "experiment network" that is isolated from the Internet and all other experiments. It is the experiment net that is shown in this topology view. You will use the control network to `ssh` into your nodes, interact with their web interfaces, etc. This separation gives you more freedom and control in the private experiment network, and sets up a clean environment for repeatable research.

10.5.4 List View

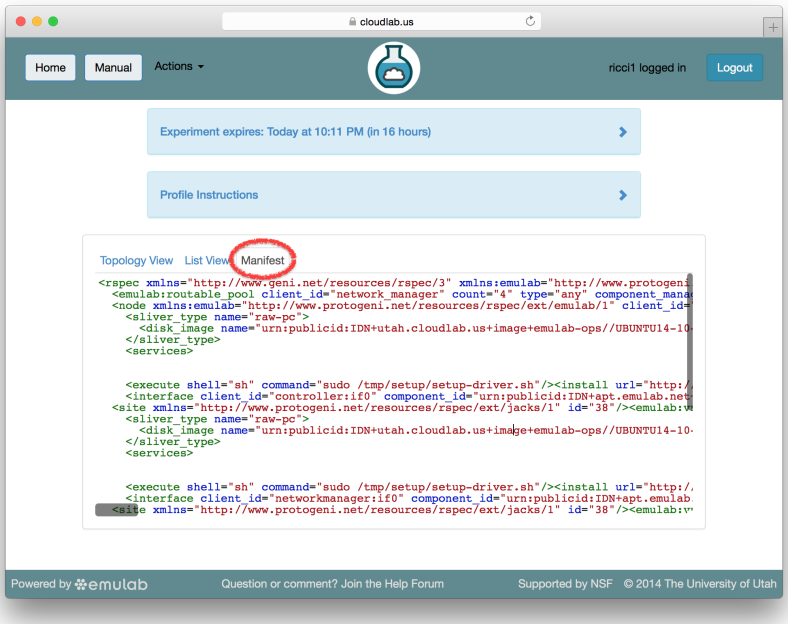
The list view tab shows similar information to the topology view, but in a different format. It shows the identities of the nodes you have been assigned, and the full `ssh` command lines to connect to them. In some browsers (those that support the `ssh://` URL scheme), you can click on the SSH commands to automatically open a new session. On others, you may need to cut and paste this command into a terminal window. Note that only public-key authentication is supported, and you must have set up an `ssh` keypair on your account **before** starting the experiment in order for authentication to work.



10.5.5 Manifest View

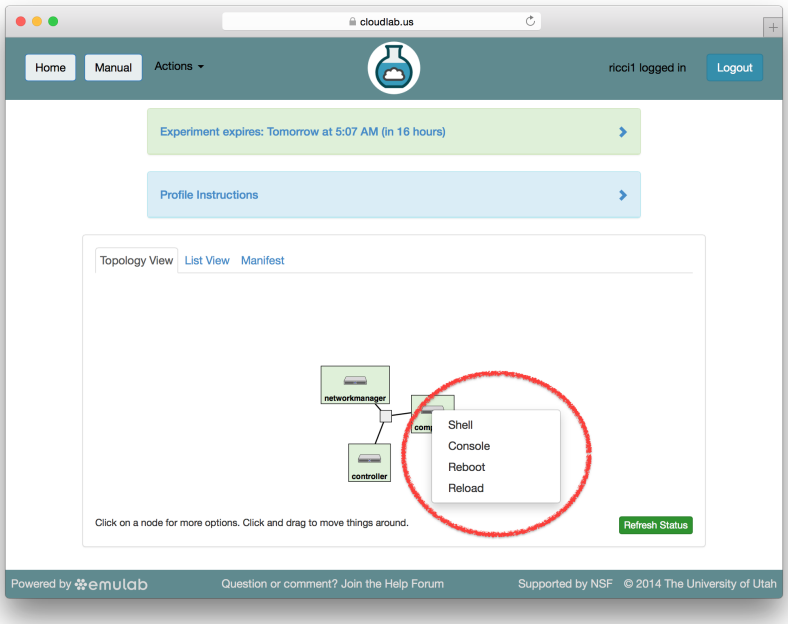
The final default tab shows a manifest detailing the hardware that has been assigned to you. This is the "request" RSpec that is used to define the profile, annotated with details of the hardware that was chosen to instantiate your request. This information is available on the nodes themselves using the `geni-get` command, enabling you to do rich scripting that is fully aware of both the requested topology and assigned resources.

Most of the information displayed on the CloudLab status page comes directly from this manifest; it is parsed and laid out in-browser.



10.5.6 Actions

In both the topology and list views, you have access to several actions that you may take on individual nodes. In the topology view, click on the node to access this menu; in the list view, it is accessed through the icon in the "Actions" column. Available actions include rebooting (power cycling) a node, and re-loading it with a fresh copy of its disk image (destroying all data on the node). While nodes are in the process of rebooting or re-imaging, they will turn yellow in the topology view. When they have completed, they will become green again. The shell and console actions are described in more detail below.



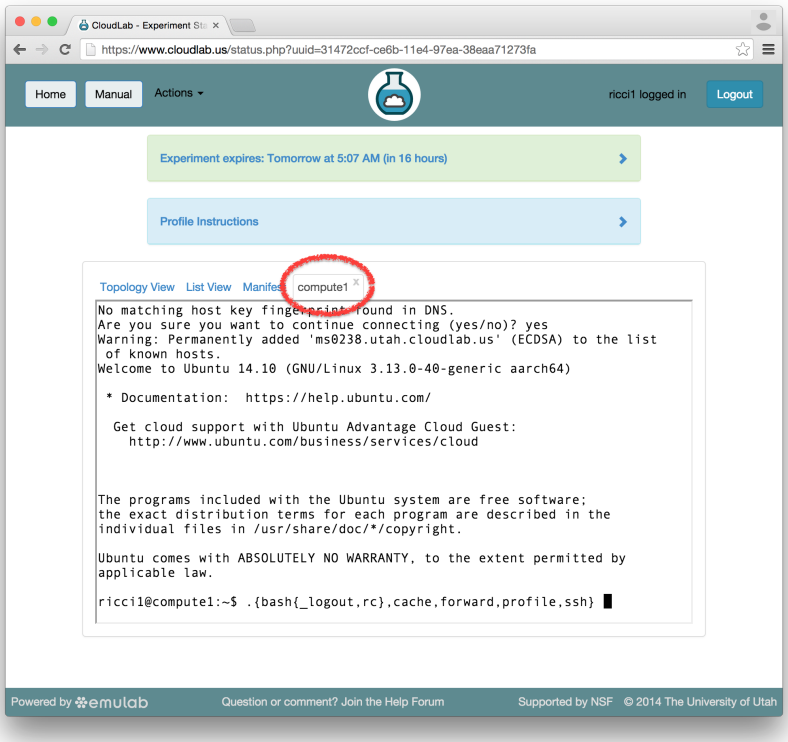
10.5.7 Web-based Shell

CloudLab provides a browser-based shell for logging into your nodes, which is accessed through the action menu described above. While this shell is functional, it is most suited to light, quick tasks; if you are going to do serious work, on your nodes, we recommend using a standard terminal and `ssh` program.

This shell can be used even if you did not establish an `ssh` keypair with your account.

Two things of note:

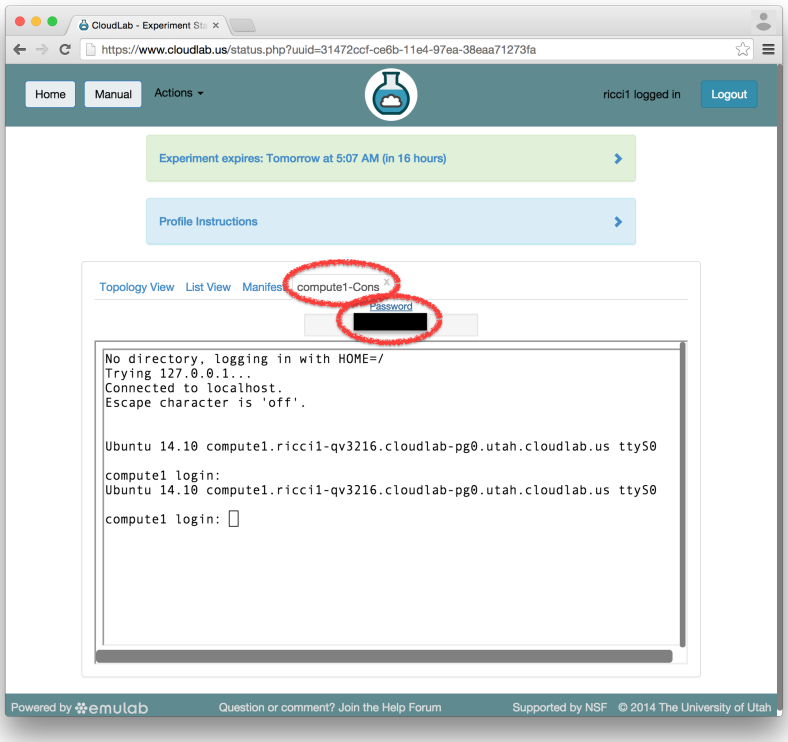
- Your browser may require you to click in the shell window before it gets focus.
- Depending on your operating system and browser, cutting and pasting into the window may not work. If keyboard-based pasting does not work, try right-clicking to paste.



10.5.8 Serial Console

CloudLab provides serial console access for all nodes, which can be used in the event that normal IP or `ssh` access gets intentionally or unintentionally broken. Like the browser-based shell, it is launched through the access menu, and the same caveats listed above apply as well. In addition:

- If you look at the console for a node that is already booted, the console may be blank due to a lack of activity; press enter several times to get a fresh login prompt.
- If you need to log in, do so as **root**; your normal user account does not allow password login. There is a link above the console window that reveals the randomly-generated root password for your node. Note that this password changes frequently for security reasons.



10.6 Bringing up Instances in OpenStack

Now that you have your own copy of OpenStack running, you can use it just like you would any other OpenStack cloud, with the important property that you have full `root` access to every machine in the cloud and can modify them however you'd like. In this part of the tutorial, we'll go through the process of bringing up a new VM instance in your cloud.

We'll be doing all of the work in this section using the Horizon web GUI for OpenStack, but you could also `ssh` into the machines directly and use the command line interfaces or other APIs as well.

1. **Check to see if OpenStack is ready to log in**

   As mentioned earlier, this profile runs several scripts to complete the installation of OpenStack. These scripts do things such as finalize package installation, customize the installation for the specific set of hardware assigned to your experiment, import cloud images, and bring up the hypervisors on the compute node(s).

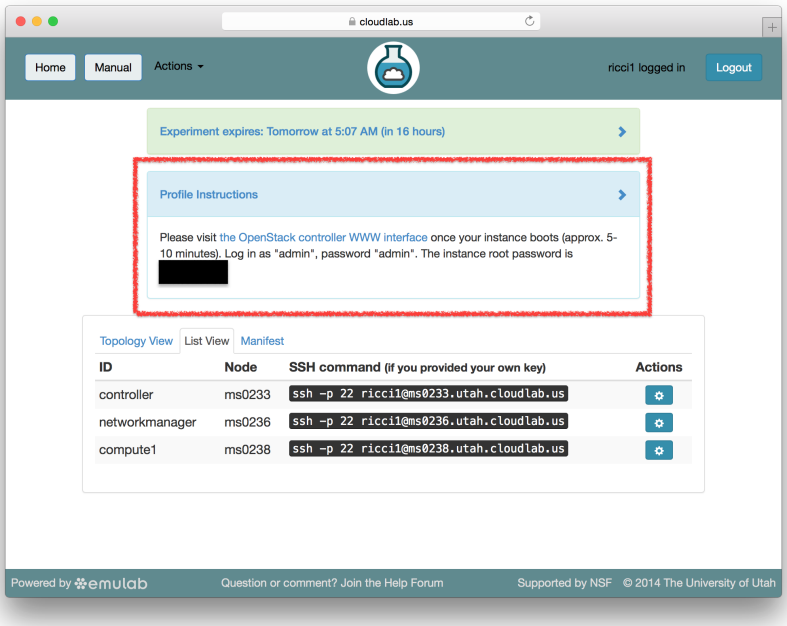
   If exploring the CloudLab experiment took you more than ten minutes, these scripts are probably done. You can be sure by checking:

- The setup script will attempt to send mail to the addresses associated with your GENI portal account when it is finished. We find, however, that this mail often gets caught by spam filters.
- You can watch the setup log by logging into the `controller` node via `ssh` (either a traditional `ssh` client, or the web-based shell described above), and running `sudo tail -F /root/setup/setup-controller.log` . When this process is finished, the final message will be something like `Your OpenStack instance has completed setup!`

If you continue without verifying that the setup scripts are complete, be aware that you may see temporary errors from the OpenStack web interface. These errors, and the method for dealing with them, are generally noted in the text below.

2. **Visit the OpenStack Horizon web interface**

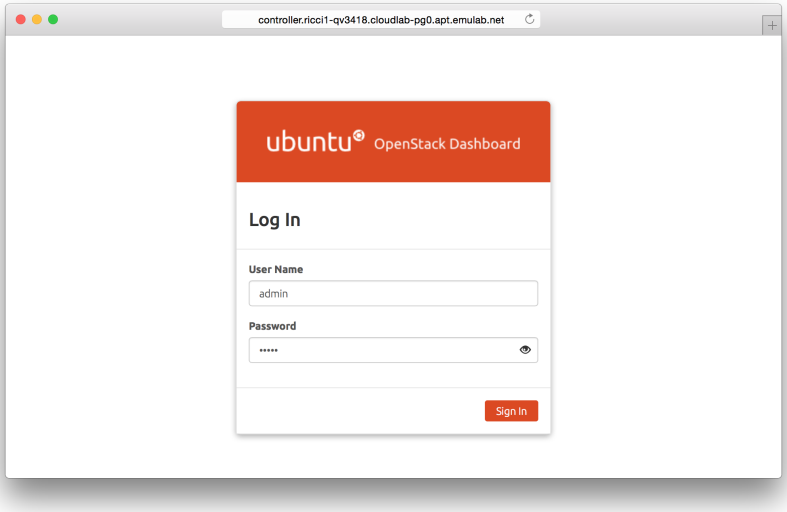
On the status page for your experiment, in the "Instructions" panel (click to expand it if it's collapsed), you'll find a link to the web interface running on the `controller` node. Open this link (we recommend opening it in a new tab, since you will still need information from the CloudLab web interface).



3. **Log in to the OpenStack web interface**

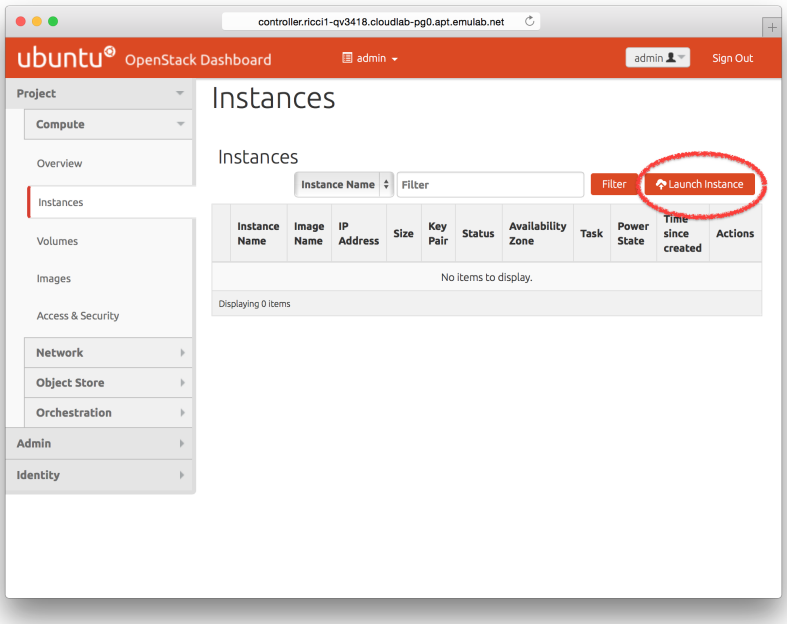
Log in using the username `admin` and the password shown in the instructions for the profile.

Important: if the web interface rejects your password, wait a few minutes and try again. If it gives you another type of error, you may need to wait a minute and reload the page to get login working properly.



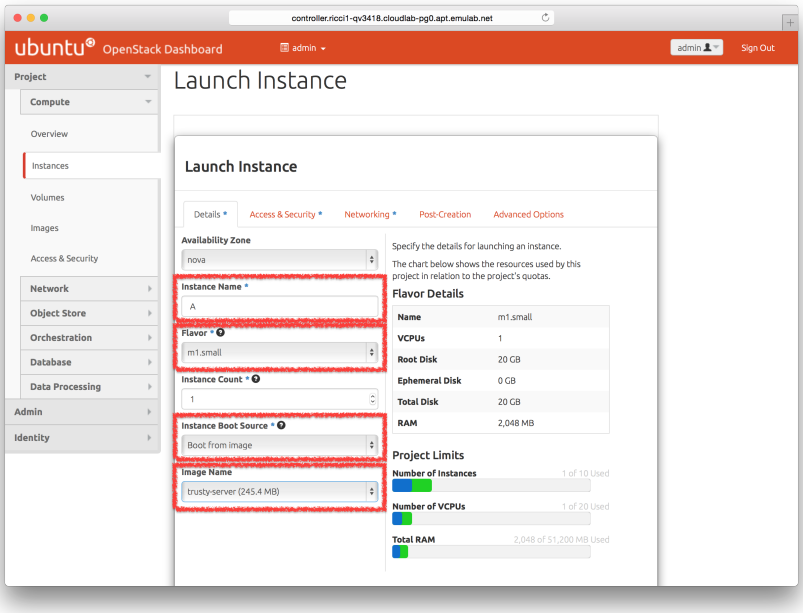
4. **Launch a new VM instance**

   Click the “Launch Instance” button on the “Instances” page of the web interface.



5. **Set Basic Settings For the Instance**

   There a few settings you will need to make in order to launch your instance:

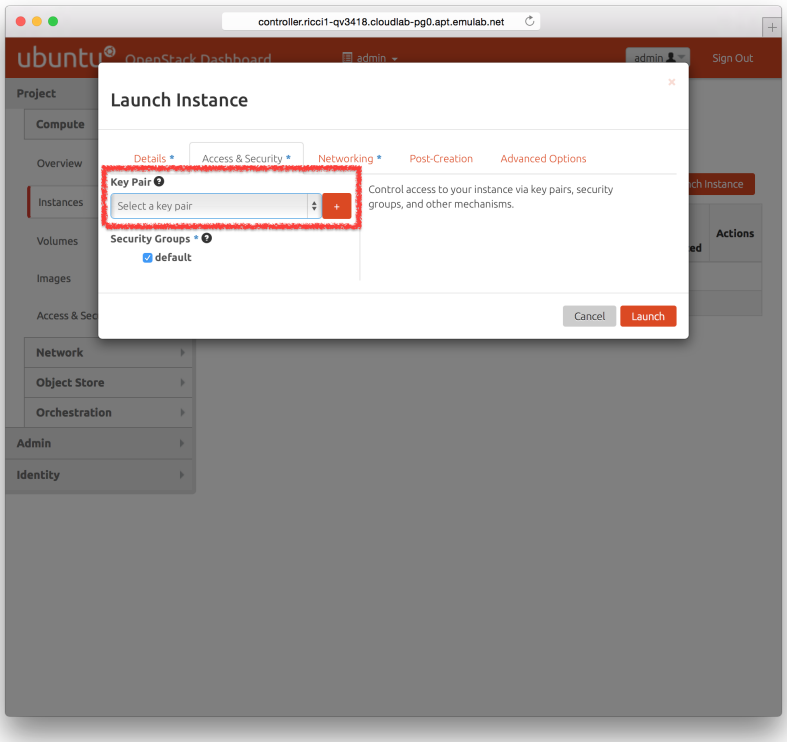


- Pick any "Instance Name" you wish
- Set the "Flavor" to `m1.small`—the disk for the default `m1.tiny` instance is too small for the image we will be using, and since we have only one compute node, we want to avoid using up too many of its resources.
- For the "Instance Boot Source", select "Boot from image"
- For the "Image Name", select "trusty-server"

Important: If you do not see any available images, the image import script may not have finished yet; wait a few minutes, reload the page, and try again.

6. **Set an SSH Keypair**

   On the "Access & Security" tab, you will add an `ssh` keypair to log into your node. If you configured an `ssh` key in your GENI account, you should find it as one of the options in this list. If not, you can add a new keypair with the "+" button. Alternately, you can skip this step and use a password for login later.

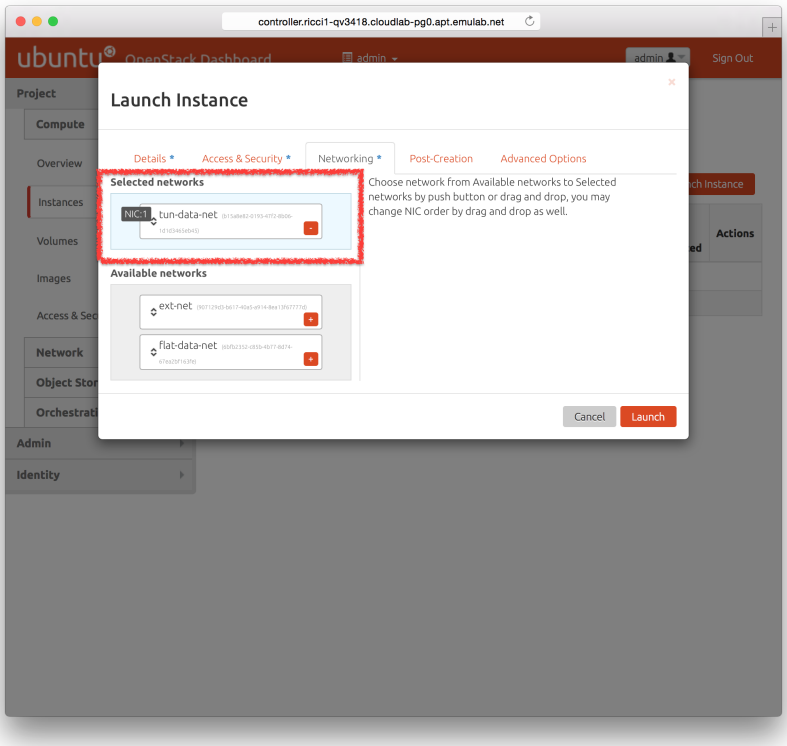


7. **Add a Network to the Instance**

   In order to be able to access your instance, you will need to give it a network. On the "Networking" tab, add the `tun-data-net` to the list of selected networks by clicking the "+" button or dragging it up to the blue region. This will set up an internal tunneled connection within your cloud; later, we will assign a public IP address to it.

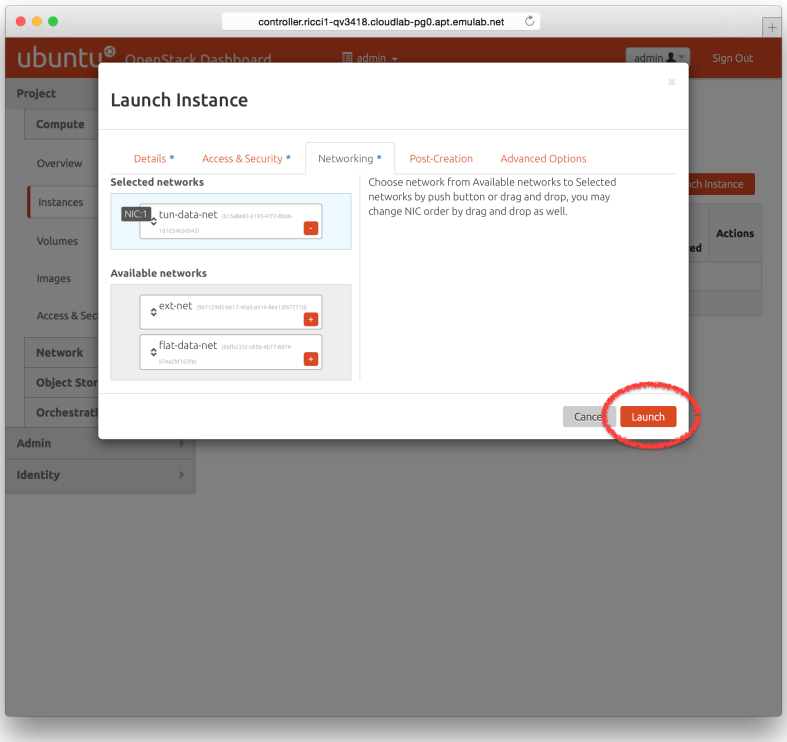
   The `tun-data-net` consists of EGRE tunnels on the CloudLab experiment network.

   Important: If you are missing the Networking tab, you may have logged into the OpenStack web interface before all services were started. Unfortunately, reloading does not solve this, you will need to log out and back in.

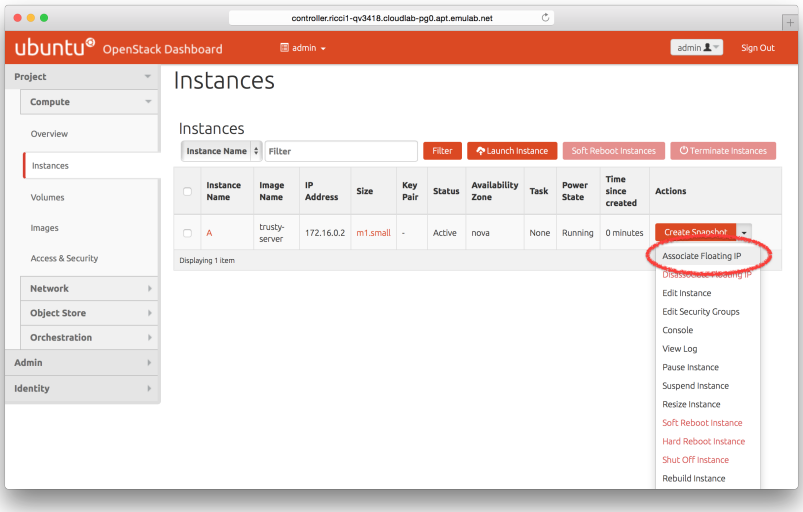


8. **Launch, and Wait For Your Instance to Boot**

   Click the “Launch” button, and wait for the status on the instances page to show up as “Active”.



9. **Add a Public IP Address**



At this point, your instance is up, but it has no connection to the public Internet. From the menu on the right, select "Associate Floating IP".

On the following screen, you will need to:

(a) Press the "+" button to set up a new IP address

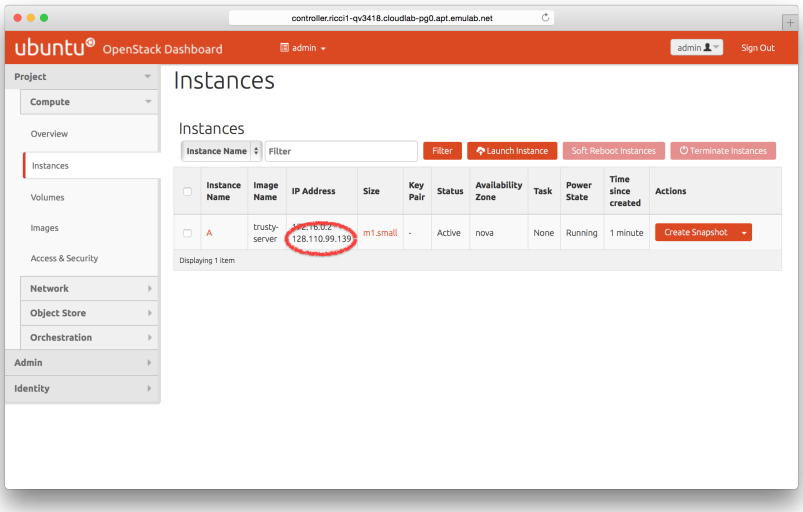
(b) Click the "Allocate IP" button—this allocates a new address for you from the public pool available to your cloud.

(c) Click the "Associate" button—this associates the newly-allocated address with this instance.

You will now see your instance's public address on the "Instances" page, and should be able to ping this address from your laptop or desktop.

Profiles may request to have public IP addresses allocated to them; this profile requests four (two of which are used by OpenStack itself.)

The public address is tunneled by the `networkmanager` node from the control network to the experiment network.



10. **Log in to Your Instance**

    You can `ssh` to this IP address. If you provided a public key earlier, use the username **ubuntu** and your private `ssh` key. If you did not set up an `ssh` key, use the username **root** and the password shown in the profile instructions. Run a few commands to check out the VM you have launched.

10.7 Administering OpenStack

Now that you've done some basic tasks in OpenStack, we'll do a few things that you would not be able to do as a user of someone else's OpenStack installation. These just scratch the surface—you can upgrade, downgrade, modify or replace any piece of your own copy of OpenStack.

10.7.1 Log Into The Control Nodes

If you want to watch what's going on with your copy of OpenStack, you can use `ssh` to log into any of the hosts as described above in the §10.5.4 "List View" or §10.5.7 "Web-based Shell" sections. Don't be shy about viewing or modifying things; no one else is using your cloud, and if you break this one, you can easily get another.

Some things to try:

- Run `ps -ef` on the `controller` to see the list of OpenStack services running
- Run `ifconfig` on the `networkmanager` to see the various bridges and tunnels that have been brought to support the networking in your cloud
- Run `sudo virsh list --all` on `compute1` to see the VMs that are running

10.7.2 Reboot the Compute Node

Since you have this cloud to yourself, you can do things like reboot or re-image nodes whenever you wish. We'll reboot the `compute1` node and watch it through the serial console.

1. **Open the serial console for compute1**

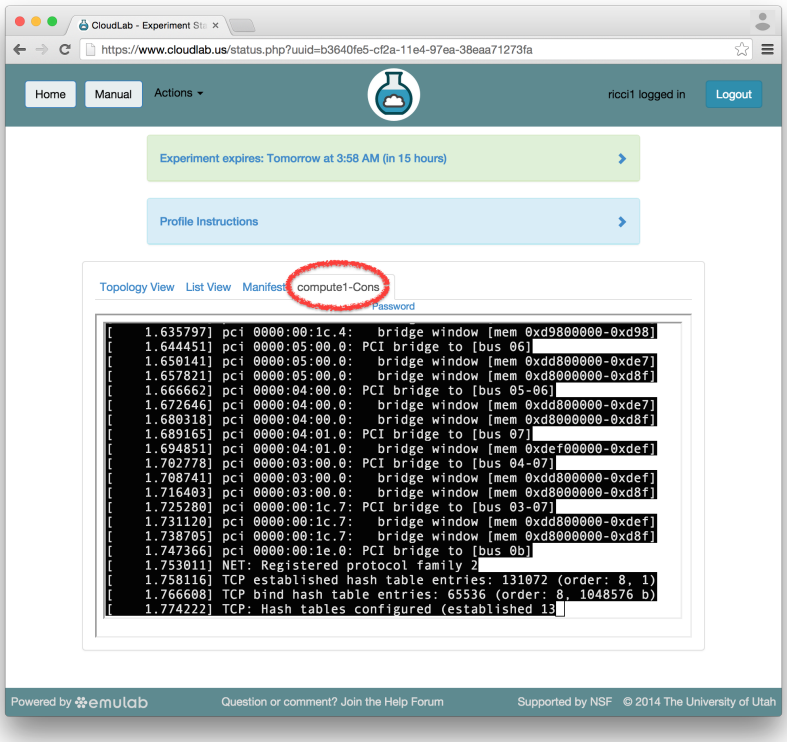
   On your experiment's CloudLab status page, use the action menu as described in §10.5.6 "Actions" to launch the console. Remember that you may have to click to focus the console window and hit enter a few times to see activity on the console.

2. **Reboot the node**

   On your experiment's CloudLab status page, use the action menu as described in §10.5.6 "Actions" to reboot `compute1`. Note that in the topology display, the box around `compute1` will turn yellow while it is rebooting, then green again when it has booted.

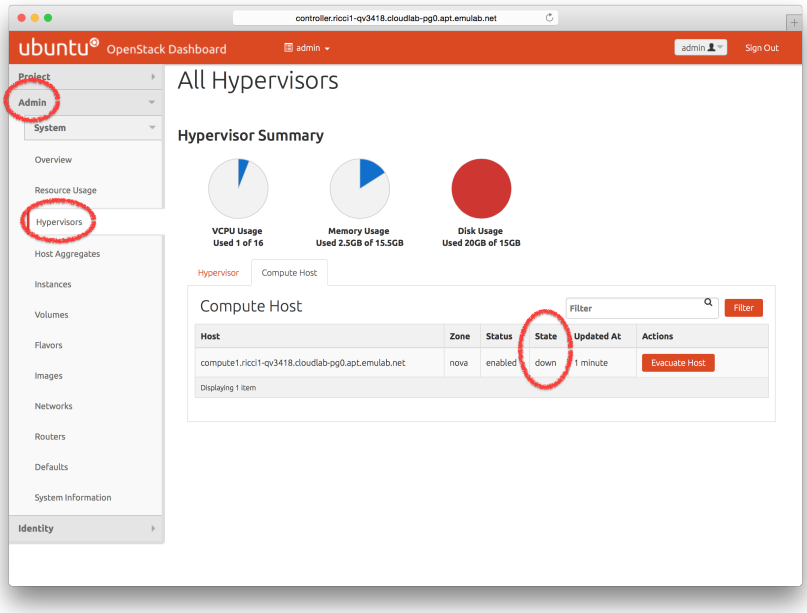
3. **Watch the node boot on the console**

   Switch back to the console tab you opened earlier, and you should see the node starting its reboot process.



4. **Check the node status in OpenStack**

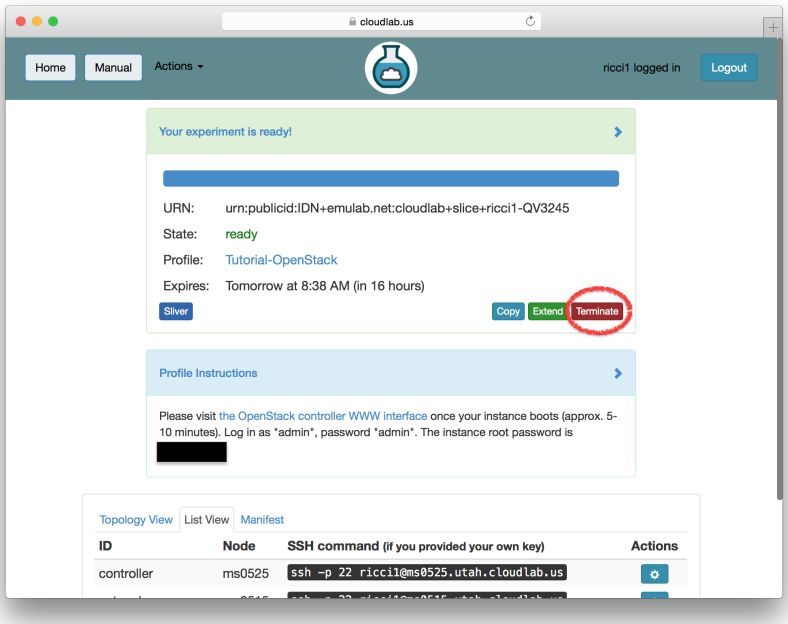
   You can also watch the node's status from the OpenStack Horizon web interface. In Horizon, select "Hypervisors" under the "Admin" menu, and switch to the "Compute Host" tab.



Note: This display can take a few minutes to notice that the node has gone down, and to notice when it comes back up. Your instances will not come back up automatically; you can bring them up with a "Hard Reboot" from the "Admin -> System ->Instances" page.

10.8 Terminating the Experiment

Resources that you hold in CloudLab are real, physical machines and are therefore limited and in high demand. When you are done, you should release them for use by other experimenters. Do this via the "Terminate" button on the CloudLab experiment status page.



Note: When you terminate an experiment, all data on the nodes is lost, so make sure to copy off any data you may need before terminating.

If you were doing a real experiment, you might need to hold onto the nodes for longer than the default expiration time. You would request more time by using the "Extend" button the on the status page. You would need to provide a written justification for holding onto your resources for a longer period of time.

10.9 Taking Next Steps

Now that you've got a feel for for what CloudLab can do, there are several things you might try next:

- Create a new project to continue working on CloudLab past this tutorial
- Try out some profiles other than OpenStack (use the "Change Profile" button on the "Start Experiment" page)
- Read more about the basic concepts in CloudLab
- Try out different hardware

- Learn how to make your own profiles

11 Getting Help

The help forum is the main place to ask questions about CloudLab, its use, its default profiles, etc. Users are strongly encouraged to participate in answering other users' questions. The help forum is handled through Google Groups, which, by default, sends all messages to the group to your email address. You may change your settings to receive only digests, or to turn email off completely.

The forum is searchable, and we recommend doing a search before asking a question.

The URL for joining or searching the forum is: `https://groups.google.com/forum/#!forum/cloudlab-users`

If you have any questions that you do not think are suitable for a public forum, you may direct them to support@cloudlab.us