Chameleon Cloud Documentation

Release 2.0a

Chameleon Cloud

Jul 01, 2026

INTRODUCTION


Chameleon

CHAPTER

ONE

ABOUT CHAMELEON

Chameleon is an NSF-funded testbed system for Computer Science experimentation. It provides researchers with deeply reconfigurable cloud infrastructure for systems, networking, distributed computing, and security research. Unlike traditional cloud services, Chameleon offers both bare metal access to physical hardware and traditional virtual machines, giving you full control over the software stack and enabling reproducible experimental research.

> **Note**
>
> **Looking for CHI@Edge? CHI@Edge's** container-based edge computing resources are documented separately — see the CHI@Edge docs

1.1 Key features

Hardware Access

- **Bare metal instances**: Full control over physical servers without virtualization overhead
- **Virtual machines**: Traditional OpenStack KVM instances for development and testing
- **Diverse hardware**: Intel/AMD CPUs (with ROCm support), ARM ThunderX2, GPUs, FPGAs, Atom processors, high-memory nodes
- **Storage options**: NVMe SSDs, traditional spinning disks, shared file systems
- **High-performance networking**: InfiniBand, 25/100 Gigabit Ethernet

Experimental Capabilities

- **Resource isolation**: Dedicated hardware reservations for reproducible experiments
- **Custom images**: Create and share disk images with your experimental software
- **Power monitoring**: Measure energy consumption at the node and application level
- **Performance metrics**: Built-in monitoring and data collection tools

Advanced Networking

- **Isolated networks**: Create private Layer-2 VLANs for multi-node experiments
- **Multi-site Layer-3**: Direct routing between Chameleon sites via FABRIC (FabnetV4)
- **WAN connectivity**: Connect to external networks and other testbeds via FABRIC
- **Flexible topologies**: Advanced routing and network configuration

Collaboration & Reproducibility

- **Trovi sharing portal**: Package and share complete experimental environments

- **Jupyter integration**: Interactive development and data analysis environment
- **Multi-site deployment**: Experiments across geographically distributed sites

1.2 Getting started

New to Chameleon?
Start with our *getting-started guide* to create an account, join a project, and launch your first instance.

Need access to the testbed?
Learn about *PI eligibility* and *project management*.

Ready to use the testbed?
Choose your interface:

- *Web Interface* - Point-and-click access to all features (**recommended for beginners**)
- *Command Line* - Programmatic access and automation
- *Jupyter Environment* - Interactive notebooks and data analysis

1.3 Quick navigation

Core Workflow

1. *Discover resources* - Find the right hardware for your experiment
2. *Make reservations* - Reserve nodes and networks
3. *Launch instances* - Deploy your experimental environment
4. *Monitor and collect data* - Measure performance and energy usage

Advanced Features

- *Custom images* - Create reproducible software environments
- *Complex deployments* - Multi-node orchestration with Heat
- *Networking* - Advanced network topologies and isolation
- *FPGA programming* - Hardware acceleration experiments
- *Package and share your work* - Publish experiments via Trovi

Data & Storage

- *Object storage* - Scalable data storage and sharing
- *Shared file systems* - NFS-mounted storage for instances
- *KVM instances* - Traditional virtual machines when needed

Getting Help

- *Help desk* - Submit tickets and view system status
- *User profile* - Manage your account settings
- *Daypass access* - Temporary access for artifact reproduction

1.4 About the infrastructure

Chameleon operates multiple sites providing different capabilities:

Core Sites:

- **CHI@TACC** (Texas): Large-scale bare metal cloud with diverse Intel/AMD hardware including GigaIO nodes
- **CHI@UC** (Chicago): Networking-focused site with specialized hardware and GPU/FPGA resources
- **CHI@NCAR** (Colorado): ARM ThunderX2 nodes for edge computing and atmospheric science research
- **CHI@Edge**: Distributed edge computing with Raspberry Pi devices (including Raspberry Pi 5) — docs
- **KVM@TACC** (Texas): Traditional OpenStack cloud

Associate Sites:

- **CHI@NRP**: National Research Platform integration
- **CHI@NU**: Northwestern University integration
- **CHI@EVL**: Electronic Visualization Laboratory (UIC) integration

The testbed serves hundreds of research projects annually, supporting publications in systems, networking, distributed computing, cybersecurity, edge computing, and atmospheric sciences.

Learn more about Chameleon and join the community at https://www.chameleoncloud.org.

CHAPTER

TWO

GETTING STARTED

Welcome to the Chameleon testbed! We're excited you're here.

In this guide, we walk through the core steps of using Chameleon resources: discovering hardware, making a reservation, configuring your instance, and connecting to it.

Not sure what Chameleon is? Read our *about page* before continuing.

Attention

Before you begin, make sure you have the following:

- A **Chameleon user account** — see our *federated authentication guide* to create one
- Membership in an **active Chameleon project** — see our *project guide* to create a project or join an existing one

Project setup can take time due to review and approval processes. Complete these steps before continuing with this guide.

At the end of this tutorial, you'll have learned how to:

- Find resources using the Hardware Discovery page and check lease availability via the *Resource Calendars*
- Make *advanced reservations* for Chameleon resources
- Configure, launch, and connect to an instance running on a bare metal server
- Orchestrate a Chameleon experiment using Jupyter and python-chi *(advanced — see our companion guide)*

Table of Contents

Ready to launch your first instance on a Chameleon **bare metal** node? In this guide, we'll use the graphical user interface (GUI) on the Chameleon portal. You can head there now in a separate browser window (https://chameleoncloud.org).

Once you've completed this guide, our *companion guide* walks through the same steps using Jupyter and python-chi, Chameleon's Python library for programmatic experiment orchestration.

Important

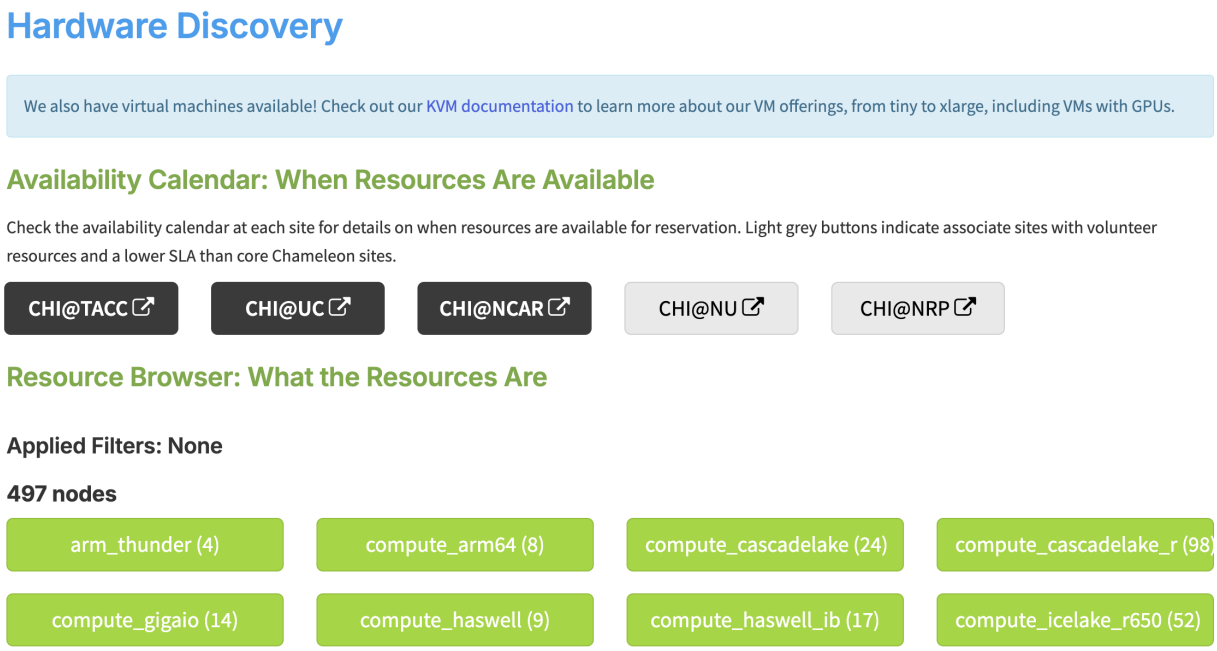
Bare metal instances are physical servers that you have exclusive access to during your reservation. This is different from virtualized clouds, where multiple users share the same physical hardware through virtualization technologies. Chameleon's bare metal approach provides users with direct access to the underlying hardware, allowing for greater customization, performance, and power monitoring. Chameleon offers bare metal reservations at CHI@TACC (Austin, TX), CHI@UC (Chicago, IL), and CHI@NCAR (Boulder, CO).

Chameleon also offers a multi-tenant, **virtualized cloud** via KVM@TACC (see *KVM*) and **container-based edge** computing via CHI@Edge (docs). See our blog for a guide on choosing between bare metal and virtualized instances.

2.1 Pick your hardware

We'll start at the Hardware Discovery page on the Chameleon Portal, where you can filter our bare metal nodes across all CHI sites using dozens of fields (including GPU configuration, memory, RAM, CPUs, etc.). You can also view detailed specifications for each node, including CPU, memory, storage, and networking.

We have our hardware pre-selected for this guide, so we'll skip most of the hardware discovery details for now. See *Resource Discovery* for a full walkthrough of the discovery tools.



Important

Chameleon resources are available *per site*, which means you **must** select the correct site to access specific hardware.

The main Chameleon Infrastructure (CHI) sites are:

- **Texas Advanced Computing Center (TACC)**: Austin, TX — `CHI@TACC`
- **University of Chicago (UC)**: Chicago, IL — `CHI@UC`
- **National Center for Atmospheric Research (NCAR)**: Boulder, CO — `CHI@NCAR`

For example, the GPU v100 node is only available at CHI@UC. Always confirm which site hosts your preferred hardware before making a reservation.

For this guide we'll use `Compute Cascadelake R` nodes, available at both CHI@UC and CHI@TACC. They're plentiful and typically reservable on demand, which makes them a good choice for a first experiment. You can follow the same steps below to get started with any bare metal hardware on Chameleon – as a challenge, try following this guide with a different node that you find on the Hardware Discovery page.

2.2 My first reservation: reserving a node

On Chameleon, you must reserve your resources before you can launch an instance on them. Chameleon supports both *on-demand* and *advanced* reservations. We will use an on-demand reservation for this guide, but note that you can reserve resources in advance, which is often necessary to get access to popular, scarce hardware like GPUs.

2.2.1 Step 1: access a testbed site

As mentioned above, different Chameleon sites have different hardware. To log in to a Chameleon site from the main Chameleon portal page, click on the `Experiment` tab on the nav bar at the top. From the dropdown, select a Chameleon site. We will be working with nodes available in either CHI@UC or CHI@TACC, so you can select one of those.

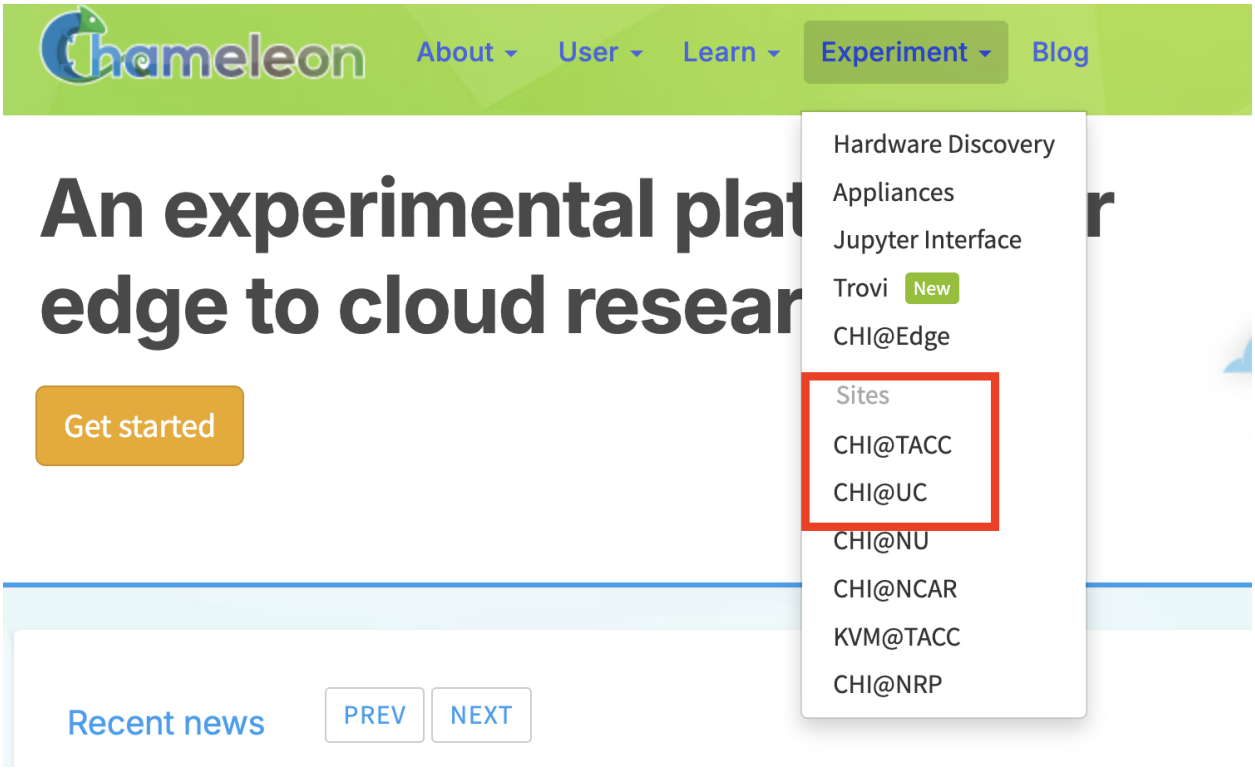


Fig. 1: Select a site to use.

When you access one of the sites, you are first taken to an **overview page**, which shows a summary of your resource usage for your current project and site. The dashboard will appear as below:

Notice that the URL has changed to a specific domain for the testbed site we chose. You can also see which site you are currently on by clicking on the dropdown next to the Chameleon logo at the top left of the window.

This section tells you which project you are currently using and which site. By clicking on the dropdown menu, you can change to another Chameleon site or change to another project.

> **Important**
>
> Projects will only appear as an option in this menu if they have a current active allocation of compute resources.

2.2.2 Step 2: go to the host calendar

From the overview page, select `Reservations` in the side navigation bar, then click `Leases`. On the Leases page, click the **Host Calendar** button.

The calendar shows a Gantt chart of node availability. Each row represents an individual node. Use the **Node Type** filter at the top to narrow the view to your desired hardware — in our case, `compute_cascadelake_r`.

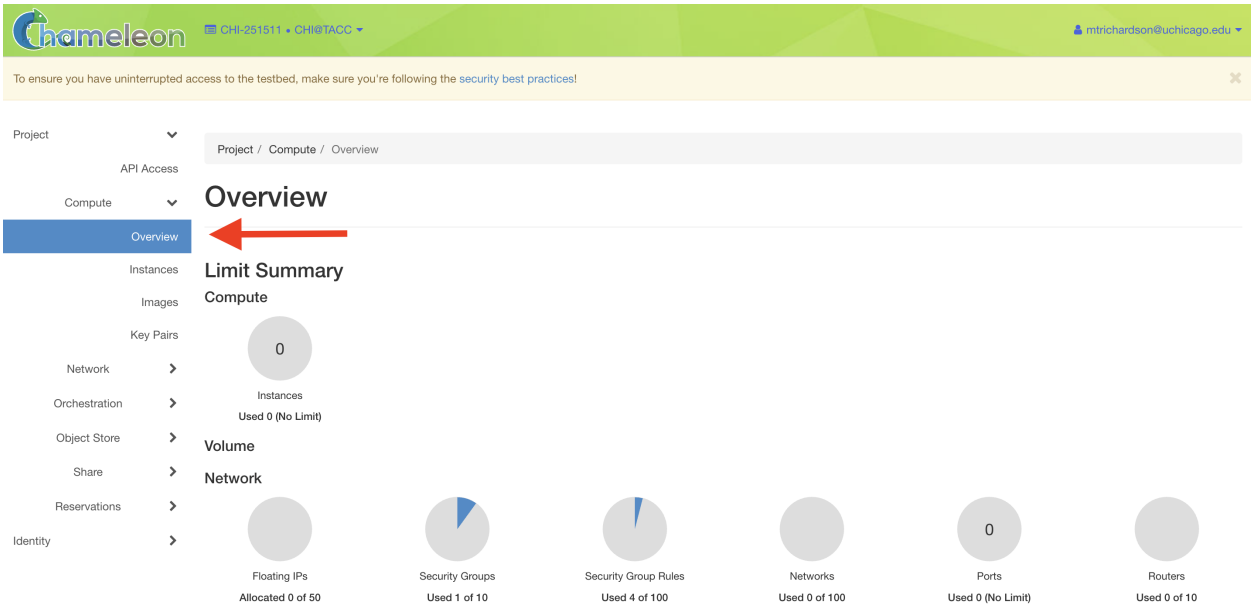


Fig. 2: An overview of your project's current resource usage for the selected site

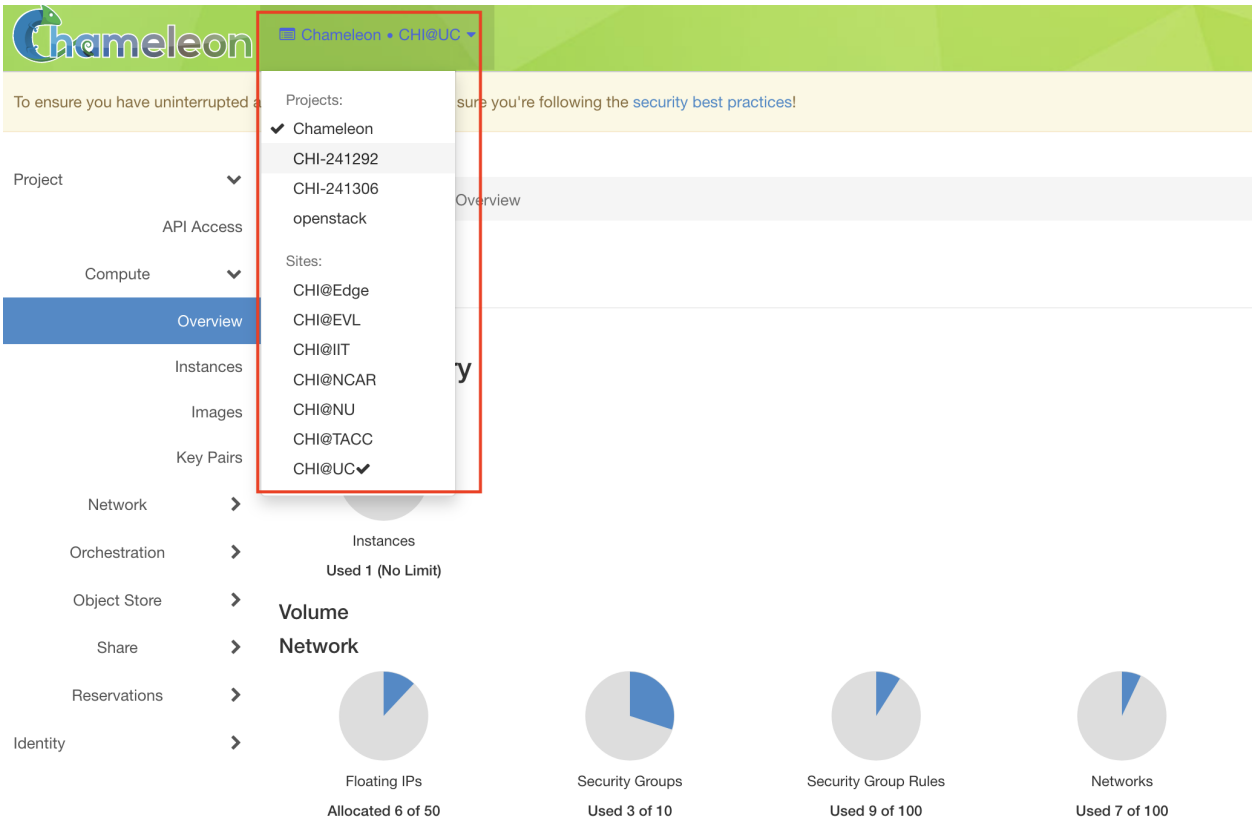


Fig. 3: Dropdown menu to change your project and site without returning to the portal

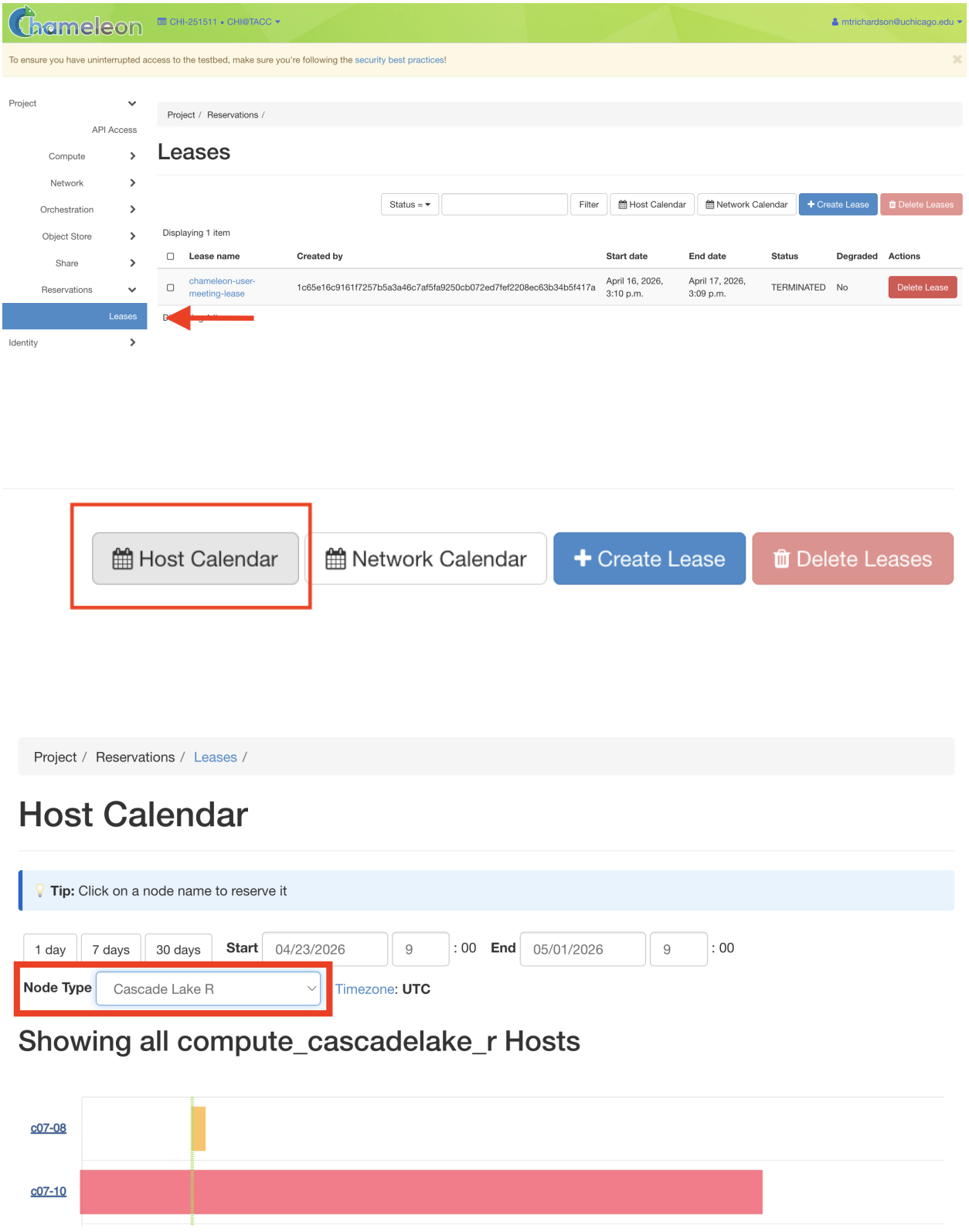


Fig. 4: Filter the calendar by node type to see which nodes are currently free.

Attention

Node types available in the filter are restricted to the site you are currently using. If you don't see a node type you expect, make sure you are logged in to the correct site.

2.2.3 Step 3: reserve a node directly from the calendar

When you spot an available node (a row with no active reservations blocking your desired start time), click on the node name – `c07-14` in the example below – associated with that row. This opens the **Create Lease** dialog with the node already pre-selected for you.

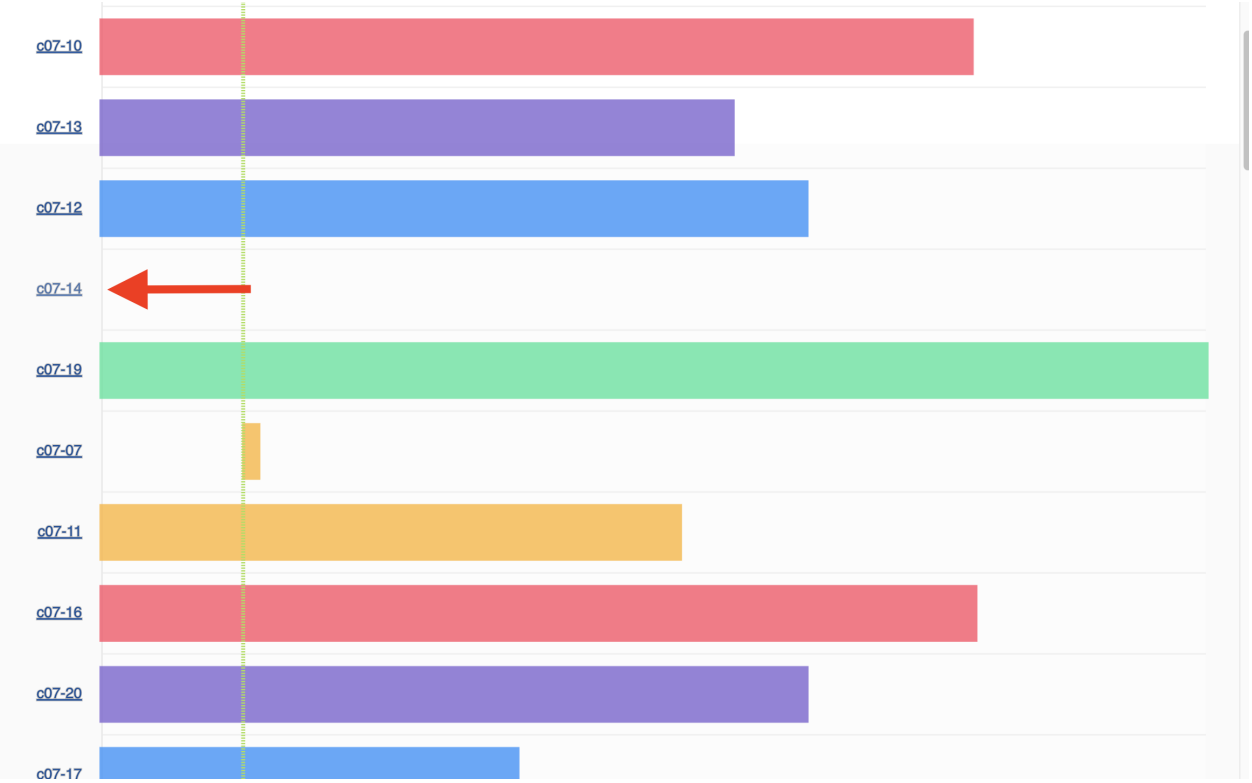


Fig. 5: Click a node row to open the Create Lease dialog with that node pre-filled.

Let's go through the lease form.

General

Give your lease a name (e.g. `my-first-lease`). For an on-demand lease you can leave the start time as-is — the form defaults to starting immediately with a one-day duration. To make an advanced reservation, adjust the start date, number of days (maximum 7), and end time.

Hosts

Because we clicked a specific node in the calendar, the Hosts section is already pre-populated with a resource property filter that matches that exact node. You don't need to change anything here.

In general, this section lets you filter by any resource property — node type, UUID, memory, GPU configuration, and more — to target the hardware you want. If you are creating a lease without using the calendar, this is where you would specify your node properties (e.g. `node_type == compute_cascadelake_r`) and the number of hosts (minimum and maximum).

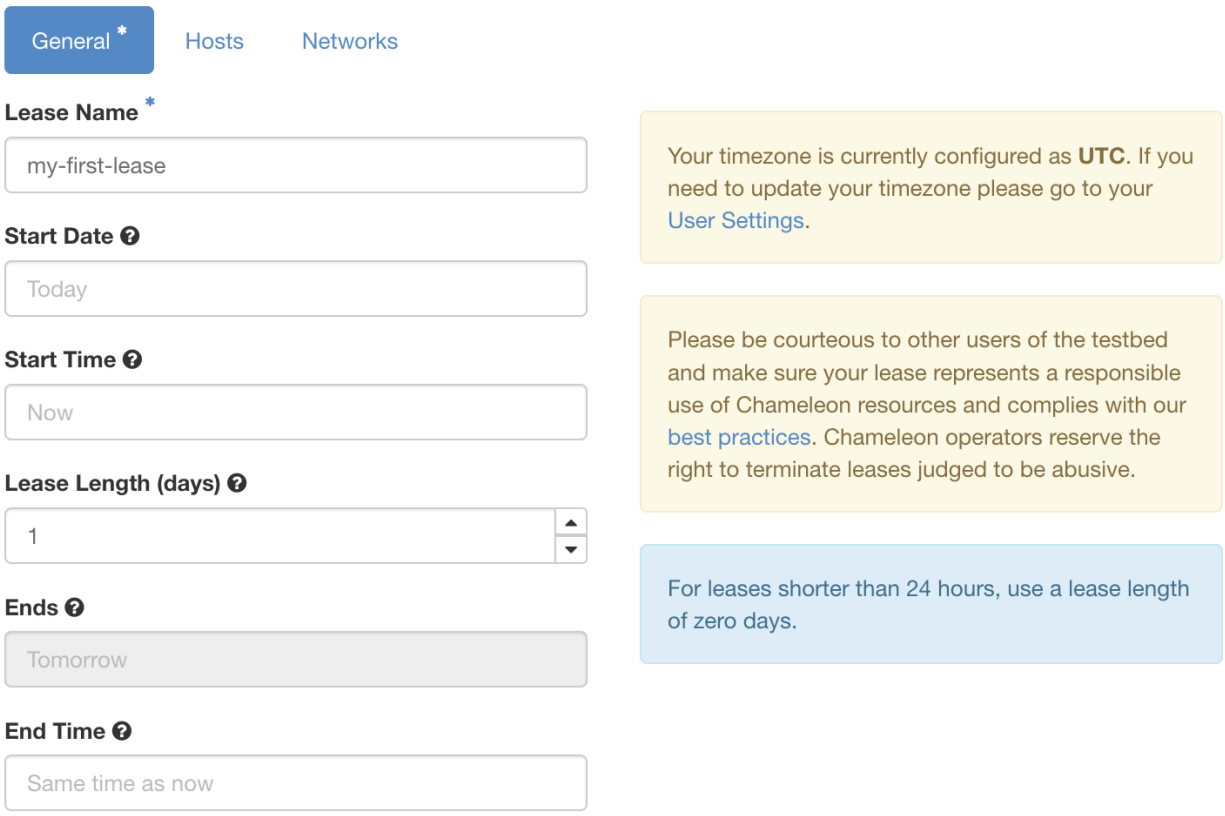


Fig. 6: Specify your lease name and duration.

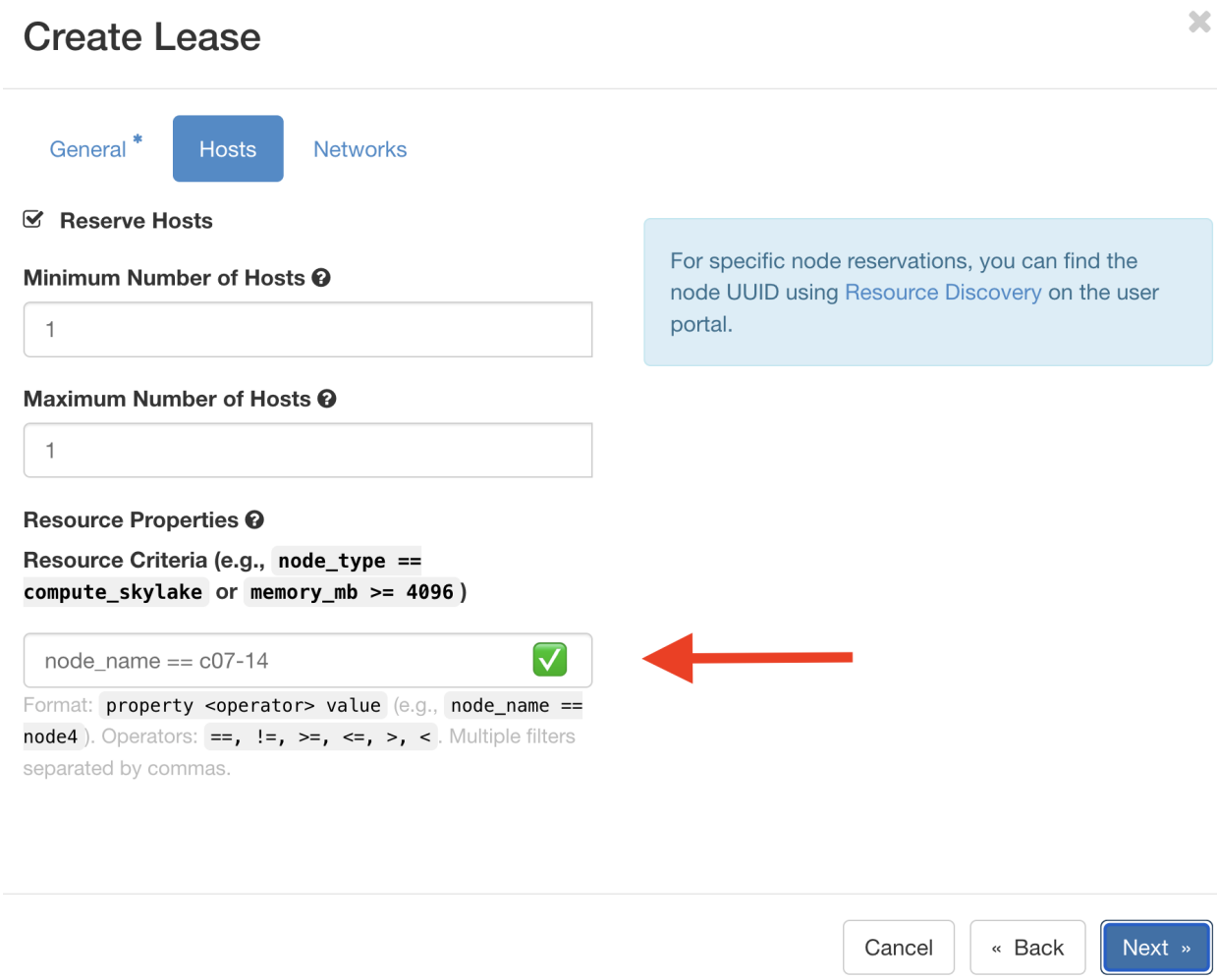


Fig. 7: The Hosts section with the node pre-selected from the calendar.

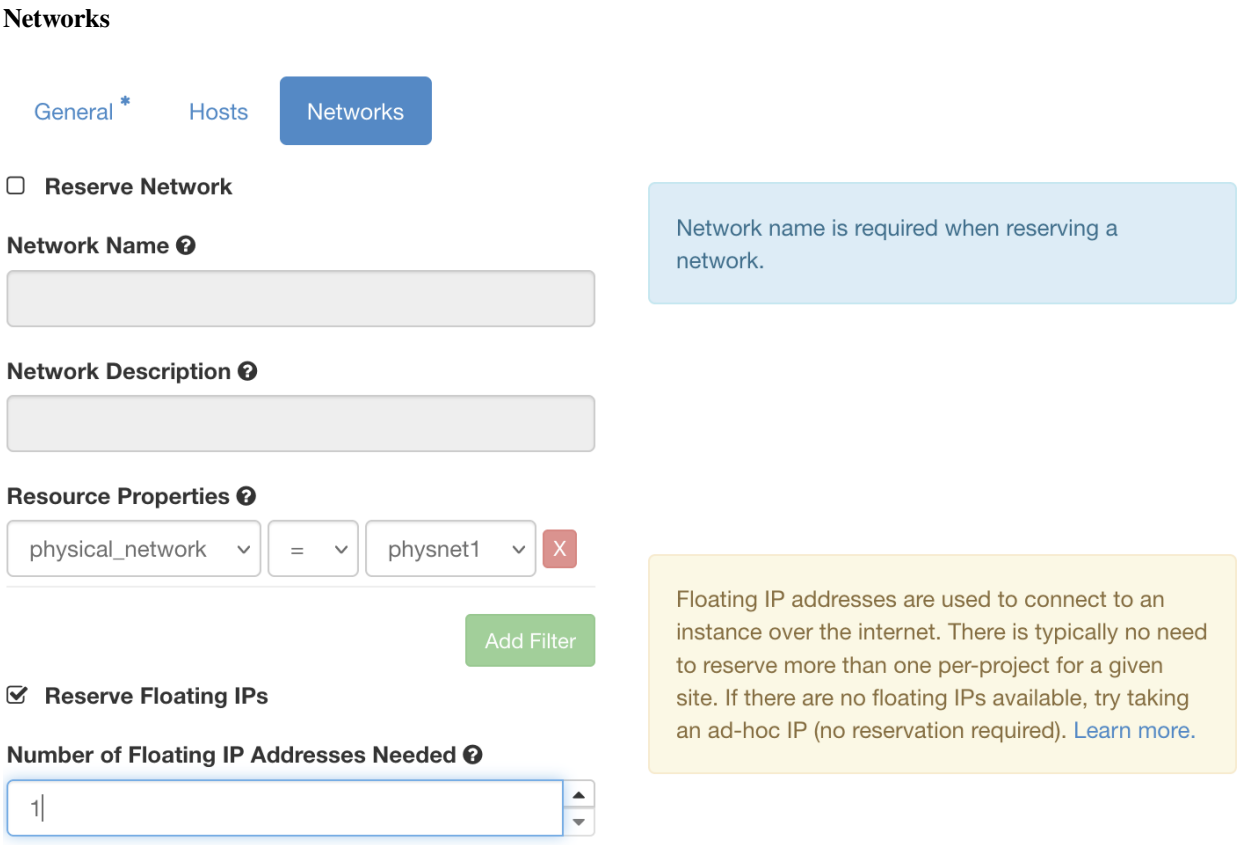


Fig. 8: Select your network reservation options.

You will almost certainly want to reserve a **Floating IP** here. Floating IPs are publicly routable addresses used to SSH into your instance from the internet. There is typically no need to reserve more than one per project per site. If no floating IPs are available in the pool, you can allocate one ad-hoc from the Floating IPs dashboard after your lease is active.

Note

Reserving a floating IP in the lease form guarantees you receive one. The pool can occasionally run dry, so reserving upfront is the safer option.

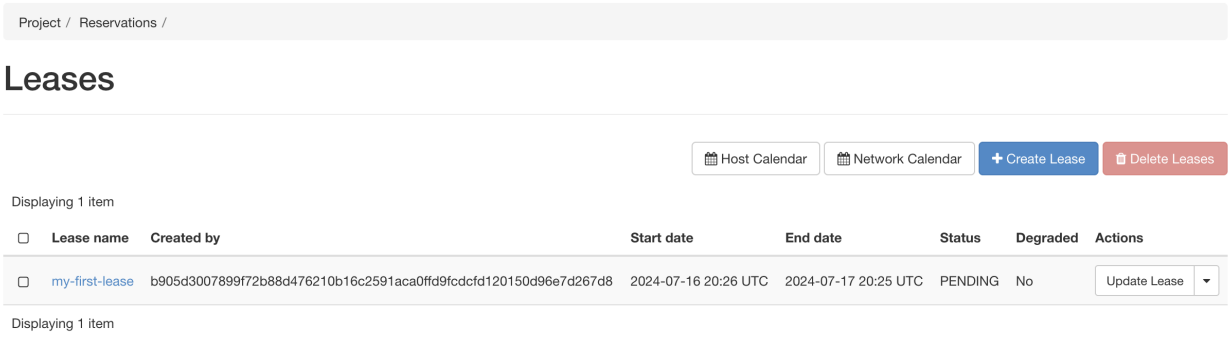
If your experiment requires an isolated private network across multiple nodes, check **Reserve Network** as well — but for a single-node first experiment, this is not necessary.

2.2.4 Step 4: submit your lease

Click **Create**. Your lease will appear on the Leases page with a status of `PENDING`. Once the scheduler confirms the reservation, it transitions to `ACTIVE` and you can launch an instance on that node.

Important

Do not stack back-to-back reservations to work around the 7-day limit. Leases found to be doing this may be deleted. See our best practices for guidance on longer reservations.



2.3 My first instance: launching an instance

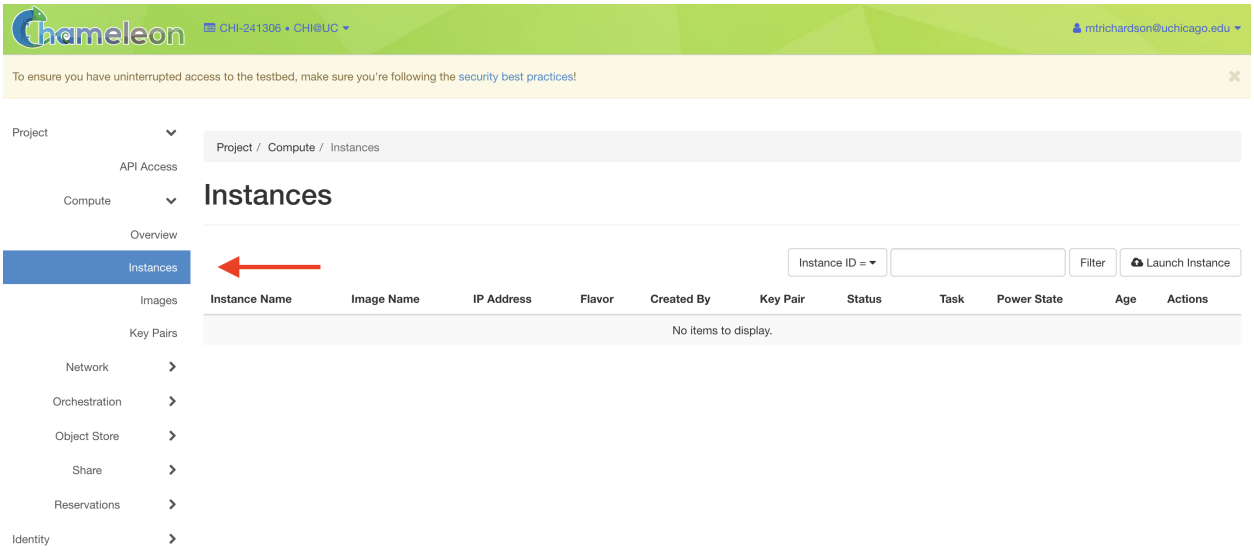
In the following steps, we will walk through how to configure and launch an instance on hardware we just reserved. In the GUI, this process will feel similar to the process we just followed to create a new lease. You will specify your instance details in a form and submit it to the system. Chameleon will then automatically configure, build, and launch your instance.

> **Note**
>
> Building and launching an instance on bare metal (especially when using beefy appliances and images) can take a long time. After creating your instance, **you may need to wait for 10 to 20 minutes before the instance will be running**.

To create a new instance, follow the steps below:

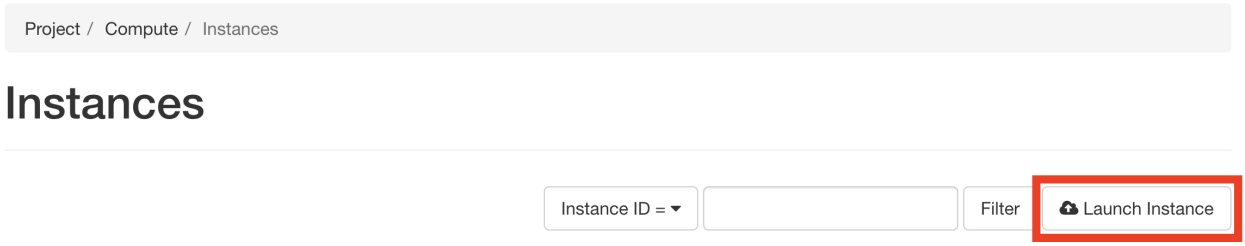
2.3.1 Step 1: go to the instances dashboard

In the sidebar from your site dasboard, click *Compute*, then click *Instances*



2.3.2 Step 2: create a new instance

Click on the *Launch Instance* button in the toolbar and the *Launch Instance* wizard will load.



Below, we go through each step of the *Launch Instance* form.

Details

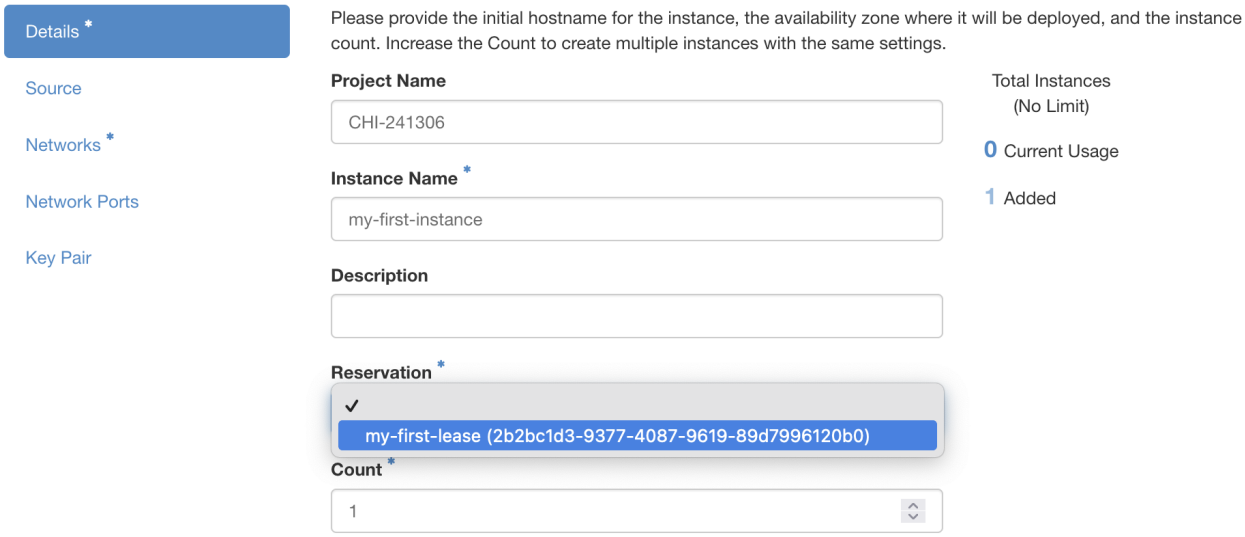


Fig. 9: Enter the main details about your instance, including which reservation to use.

Give your instance a descriptive name (*my-first-instance*) and a short description (optional). You will also need to specify the lease that you will use for this instance. You can select the reservation that you just created from the dropdown. We can also specify how many instances we want to launch. The default is one and we have one node so we'll stick with that.

Source

In the next section, we can configure a source that we will use for our instance. This can be an image, a snapshotted image, a volume, or some other appliance. Chameleon staff maintain a set of supported images (identified with a Chameleon badge in the list); these are also browsable via the Appliances Catalog on Trovi. There are also user-uploaded images and appliances. For our demo, we'll use the supported `CC-Ubuntu26.04` image. Scroll down in the list or type the name of the image in the search bar, find the image, and click the up-arrow icon next to it to allocate it as the instance source. See *Images* for full documentation on images and appliances.

Networks

On the next section, we can allocate a network to provide communication channels for instances in the cloud. For this guide, we'll use `sharednet1`, Chameleon's shared default network — it provides internet connectivity out of the box and is the simplest option for a first experiment.

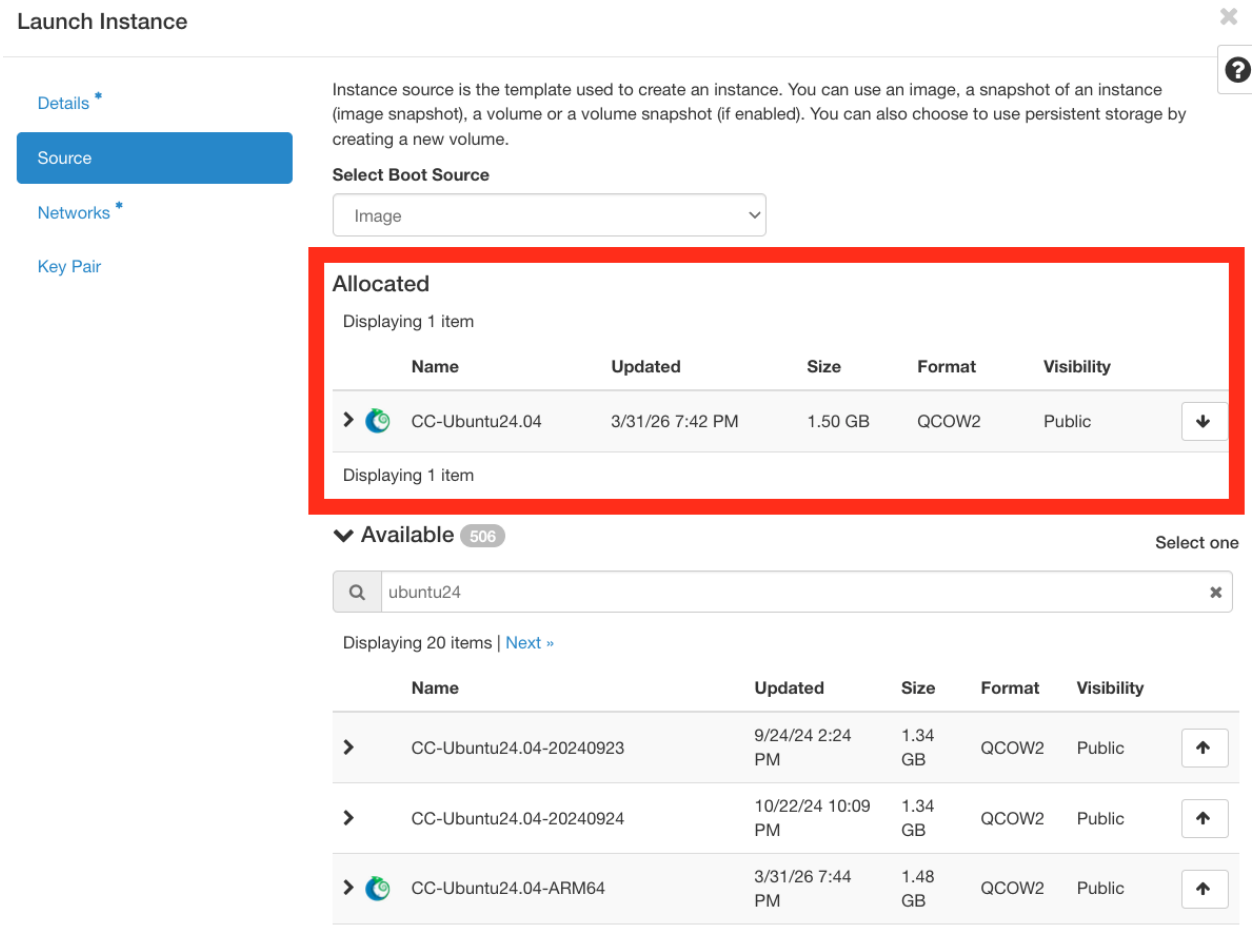


Fig. 10: Select your image source.

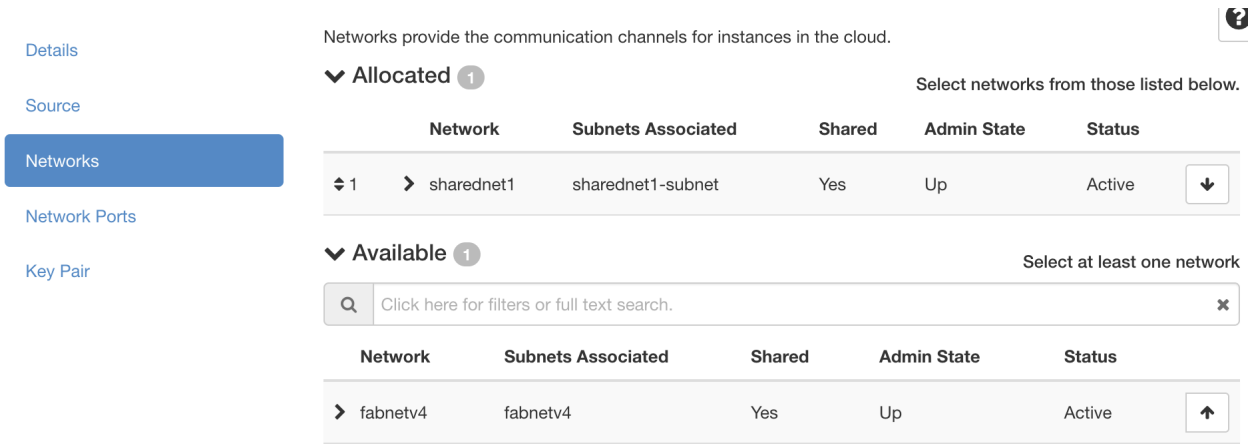


Fig. 11: Allocate a network.

`fabnetv4` is a second shared network specifically for accessing FABRIC testbed resources and cross-site stitching — see *External layer 3 connectivity* for details.

> **Note**
>
> If you plan to run many instances (for example, a class experiment with many students), consider creating a **project network** (an isolated VLAN) instead of placing all instances on `sharednet1`. A project network gives your experiment a private broadcast domain while still allowing you to assign floating IPs to individual instances for public access. See *Isolated network VLANs* for details.

Key Pairs

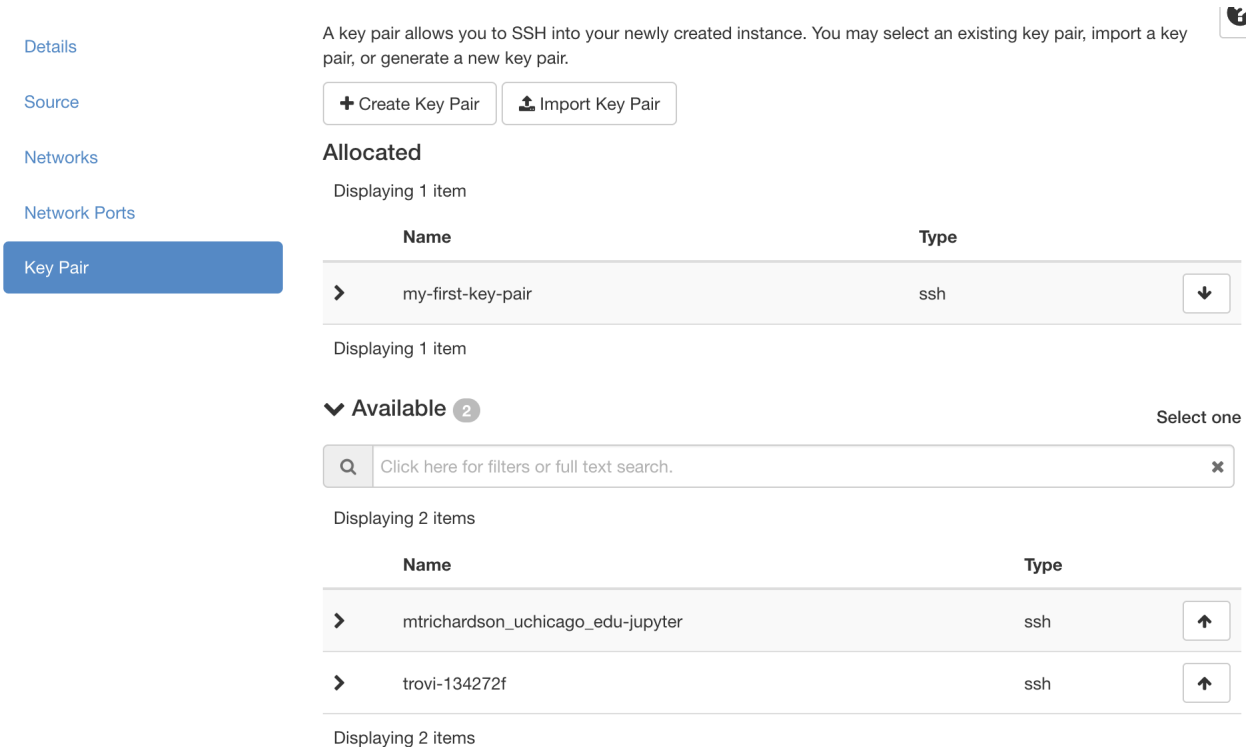


Fig. 12: Add a key pair to the instance.

As a final step to create our instance, we can set up a key pair. We **must** add a key pair if we want to remotely access the instance after it is running.

> **Tip**
>
> For a **comprehensive walkthrough of setting up a new key pair for Chameleon**, see reference here.

To add a key pair, we can either add a new one using `Create Key Pair` and storing the credentials on our local machine, or import an existing key using the `Import Key Pair`. If you have previously uploaded a key pair to Chameleon, this key pair will appear in the "Available" section below. You can then reuse that key pair.

2.3.3 Step 3: click launch

Finally, we are ready to click `Launch Instance`. Doing so will take us back to our Instances page, where we should see a new row for the instance that we just created. We can see most of the important information about our instance from here. We can also click on the instance name (like with a lease) to view more details.

The detailed page gives you an overview of the instance. There are also other options to view logs, open a console (once the instance is running), and more.

2.4 First contact: associating an IP address & SSH

Your instance may take approximately ten to fifteen minutes to launch depending on the node type. The launch process includes powering up, loading the operating system over the network, and booting up for the first time on a rack located at one of our hardware sites, depending on where you chose to launch your instance.

Once the instance is running, we will connect to it from our local computer via SSH with the key pair that we added during our configuration step. The steps in the following section will show you how to connect to your instance so you can start working with the node you reserved.

2.4.1 Step 1: associate an IP

Before you can access your instance, you need to first assign a floating IP address - an IP address that is accessible over the public Internet. You can do this from the same GUI dashboard where you made your reservation and launched your instance.

To associate an IP address with your instance, follow these steps.

1. Go to the *Floating IP* dashboard by clicking on *Network* and *Floating IPs* in the sidebar.

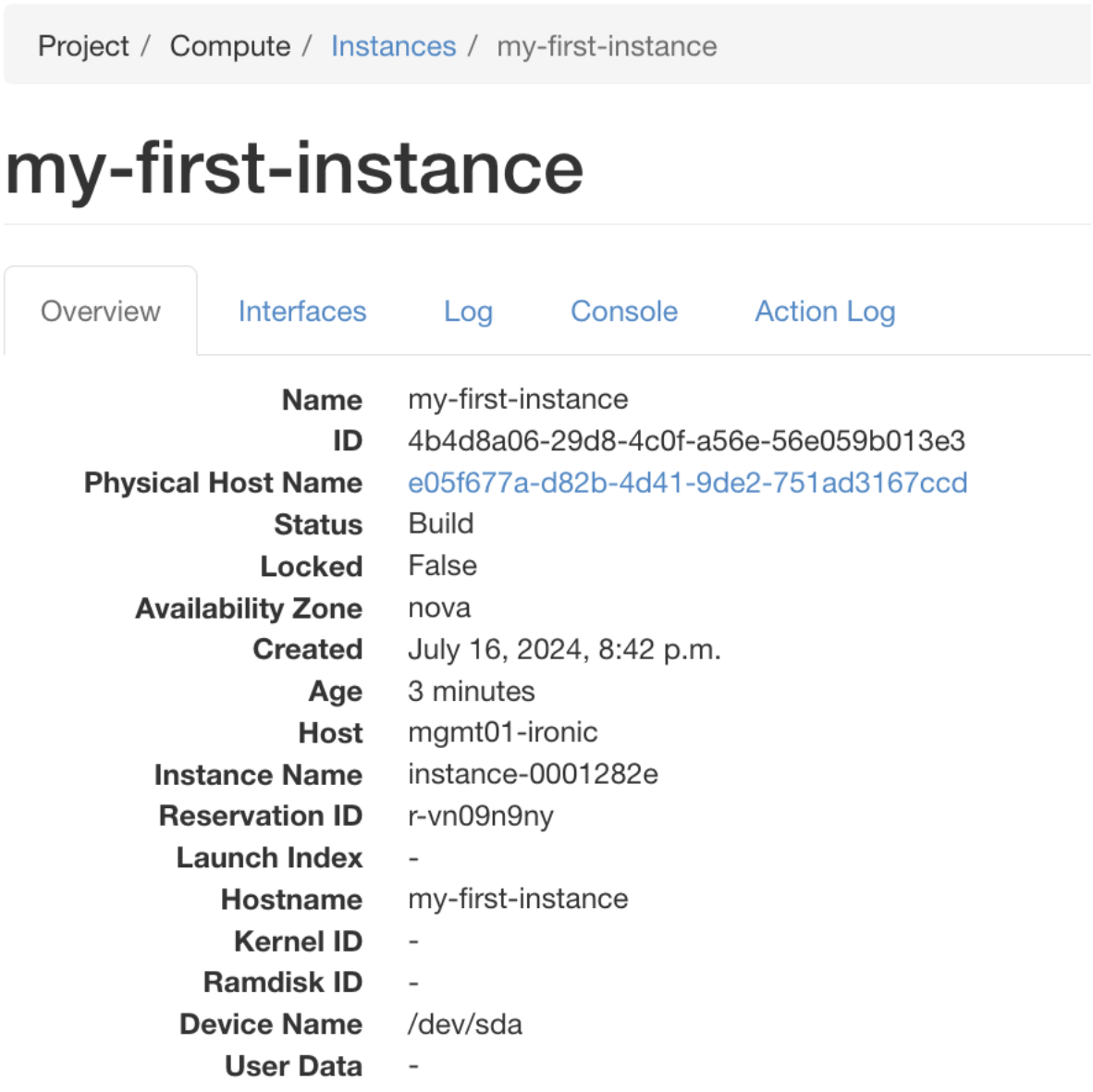
2. Look for a free Floating IP not currently associated to an instance, click the *Associate* button for the IP. A dialog will load that allows you to assign a publicly accessible IP to your instance. Click the *Associate* button in the dialog to link the public IP to your instance.

3. If you didn't already have a Floating IP available, you may allocate one to your project by clicking on the *Allocate IP to Project* button along the top row in the Floating IP dashboard. A new dialog will open for allocating the floating IP.

   Click the *Allocate IP* button. The Floating IP dashboard will reload and you should see your new Floating IP appear in the list. You can now go back to step 2.

2.4.2 Step 2: connect via SSH

Once your instance is running and has a floating IP associated, open a terminal and connect using the private key you added during the launch step. All Chameleon instances use `cc` as the default login username.


Project / Compute / Instances / my-first-instance
my-first-instance
Overview
Interfaces
Log
Console
Action Log
Name my-first-instance
ID 4b4d8a06-29d8-4c0f-a56e-56e059b013e3
Physical Host Name e05f677a-d82b-4d41-9de2-751ad3167ccd
Status Build
Locked False
Availability Zone nova
Created July 16, 2024, 8:42 p.m.
Age 3 minutes
Host mgmt01-ironic
Instance Name instance-0001282e
Reservation ID r-vn09n9ny
Launch Index -
Hostname my-first-instance
Kernel ID -
Ramdisk ID -
Device Name /dev/sda
User Data -

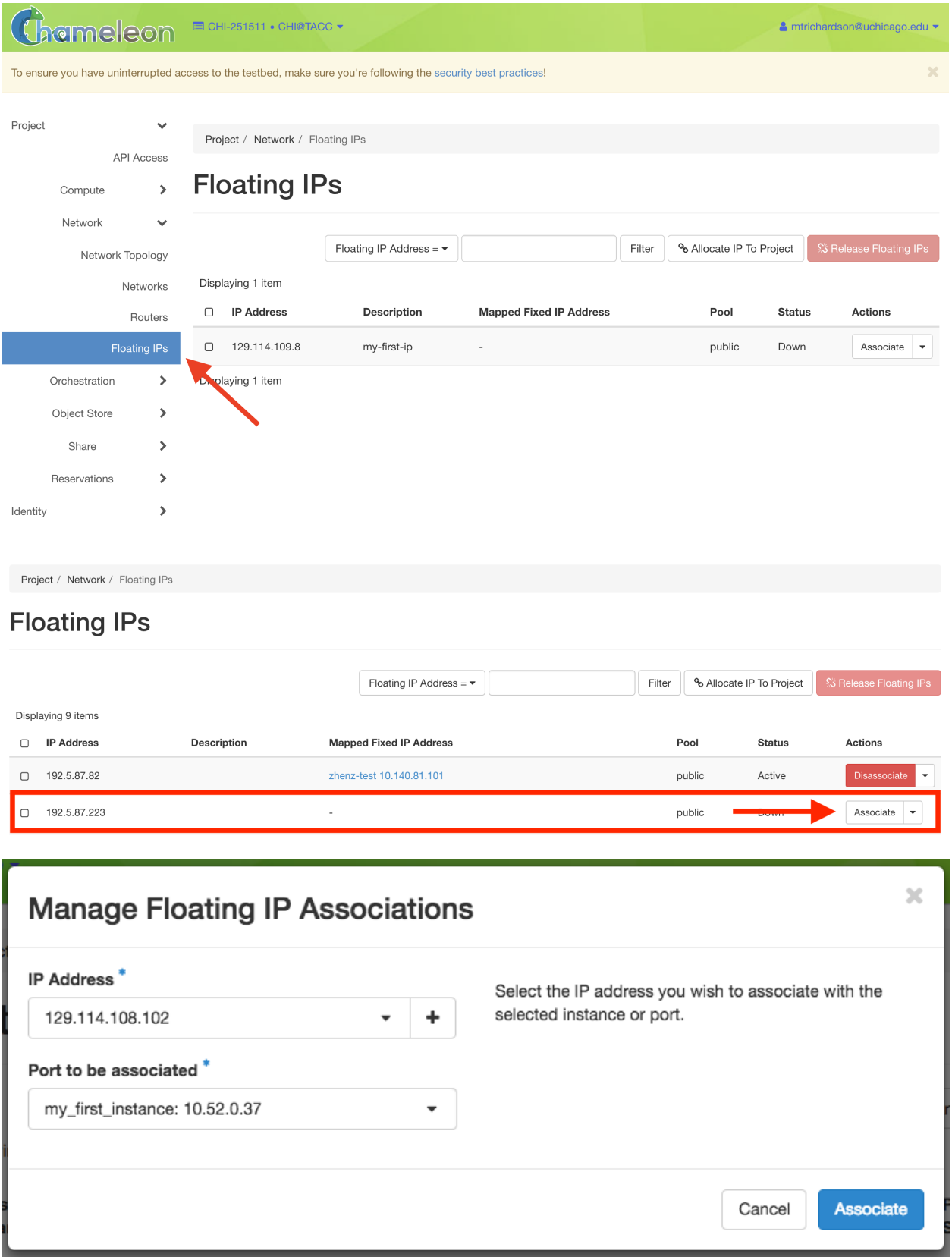


Fig. 13: Here you can assign a floating IP address

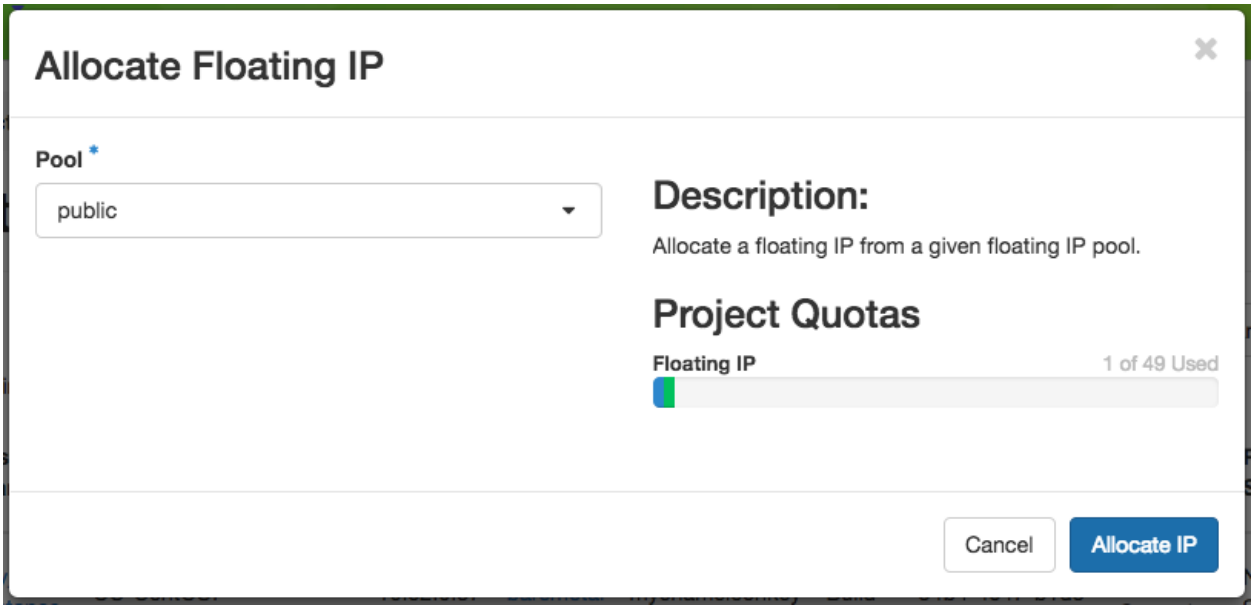


Fig. 14: This dialog allows you to allocate an IP address from Chameleon's public IP pool

Note

The instructions below assume macOS or Linux. Windows users can follow our YouTube video on logging in via SSH on Windows.

```
ssh -i /path/to/your/private/key cc@<your-floating-ip>
```

Replace `/path/to/your/private/key` with the path to your private key file and `<your-floating-ip>` with the floating IP you associated in the previous step.

> **Tip**
>
> **New to SSH keys?** See this SSH key guide for a walkthrough on generating and using key pairs.

Once connected, your terminal prompt changes to show the instance hostname, confirming you are now running commands directly on the bare metal node:

```
cc@my-first-instance:~$ lscpu
Architecture:        x86_64
CPU(s):              96
Model name:          Intel(R) Xeon(R) Gold 6240R CPU @ 2.40GHz
Thread(s) per core:  2
Core(s) per socket:  24
Socket(s):           2
```

Congratulations! You just created your first Chameleon instance!

2.5 What's next?

Now that you've reserved a node, launched an instance, and connected via SSH, you're ready to start running experiments on Chameleon.

- **Go programmatic**: Our *JupyterHub and python-chi guide* walks through the same workflow using Chameleon's Python library inside a Jupyter Notebook — no GUI required. Prefer the command line? See the *CLI guide* for scripting and automation via the OpenStack client.
- **Explore more hardware**: Browse Hardware Discovery and the *reservations guide* to learn about advanced reservation options.
- **Browse experiment templates**: Check out tutorials on Trovi for ready-made experiment patterns you can launch and adapt.

- **Watch and learn**: Visit our webinar page for live tutorials and recorded walkthroughs.

If you have questions, *see our documentation on getting help*. For feedback on this guide, reach out at contact@chameleoncloud.org.

CHAPTER

THREE

NEXT STEPS: JUPYTERHUB AND `python-chi`

In the *getting started guide*, we walked through how to find hardware, reserve resources, and launch an instance using the Chameleon web interface. In this guide, we'll accomplish the same thing programmatically using a Jupyter Notebook connected to the testbed.

3.1 Jupyter on Chameleon

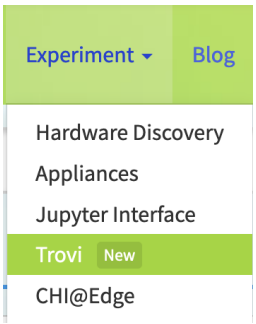
Chameleon is integrated with *JupyterHub*, so you can launch a Jupyter server (on KVM) with an environment pre-configured with python-chi and authentication to the testbed. JupyterHub on Chameleon allows you to create Jupyter Notebooks with your experiment and analysis code, collaborate with other project members in a common testbed workspace, and share files as Trovi artifacts with the Chameleon community.

To read more about the Jupyter interface, see *our docs* on the interface.

To launch the Jupyter interface on Chameleon, go to the Chameleon home page, click on the "Experiment" tab, and select the "Jupyter Interface" item. This will launch a new window which will begin loading the Jupyter server. It will then launch the JupyterHub interface. This interface should be familiar if you've ever worked with Jupyter tools before. From the launch page, we can create new notebooks, open consoles, and even open a terminal.

The work that you do in this space is persistent, so if you create a new notebook and then exit the interface and relaunch it, the notebook will still appear in your file system.

You can also download and import files from Jupyter as well as integrate with git.



3.2 Trovi

One benefit of having an interface like Jupyter available is that users can use it to package their project materials, scripts, code, and datasets as artifacts that others can replicate and extend. So, how does Chameleon facilitate this sharing?

Chameleon provides the *Trovi* service as a repository to share and access artifacts from other users on the testbed. Trovi is integrated with the Jupyter Interface, so you can launch Trovi artifacts directly onto the Jupyter Interface and start using them. You can also take your Jupyter artifacts and upload them to Trovi from Jupyter, allowing others to see and use them.

To get to the Trovi repository from the Chameleon home page, go to the "Experiment" tab and click the "Trovi" menu item. Here, you can see all the public artifacts available on the testbed.

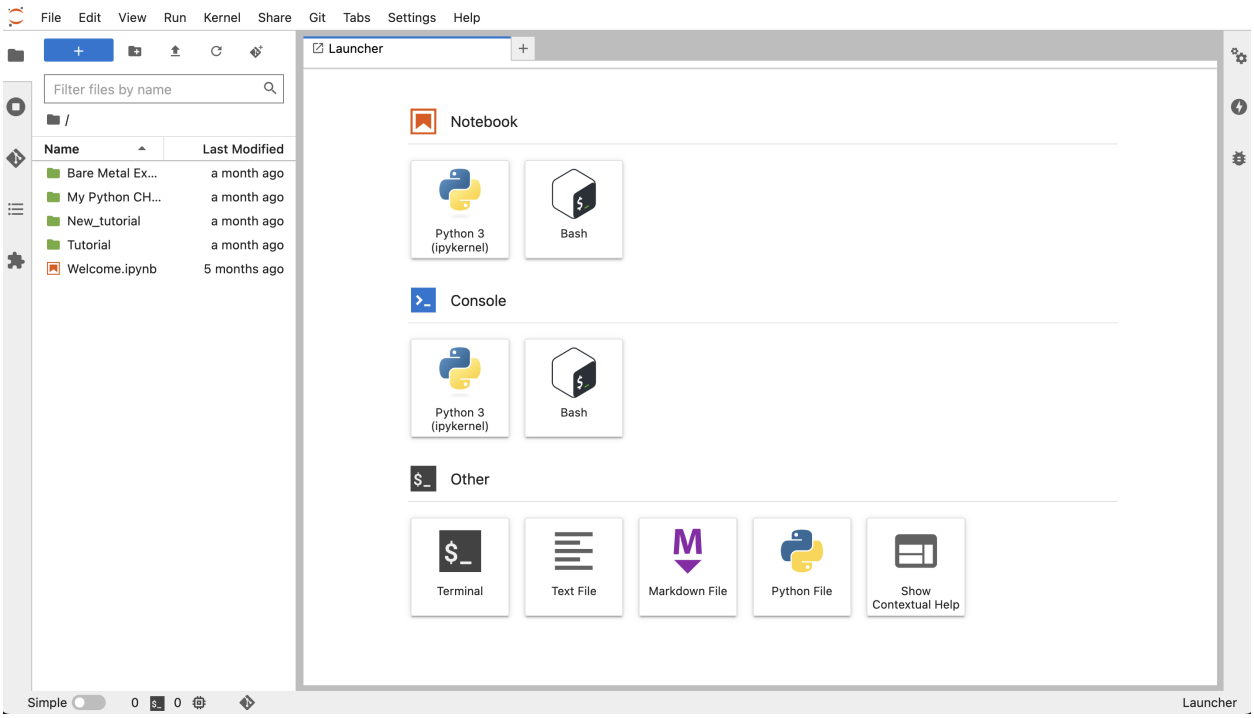
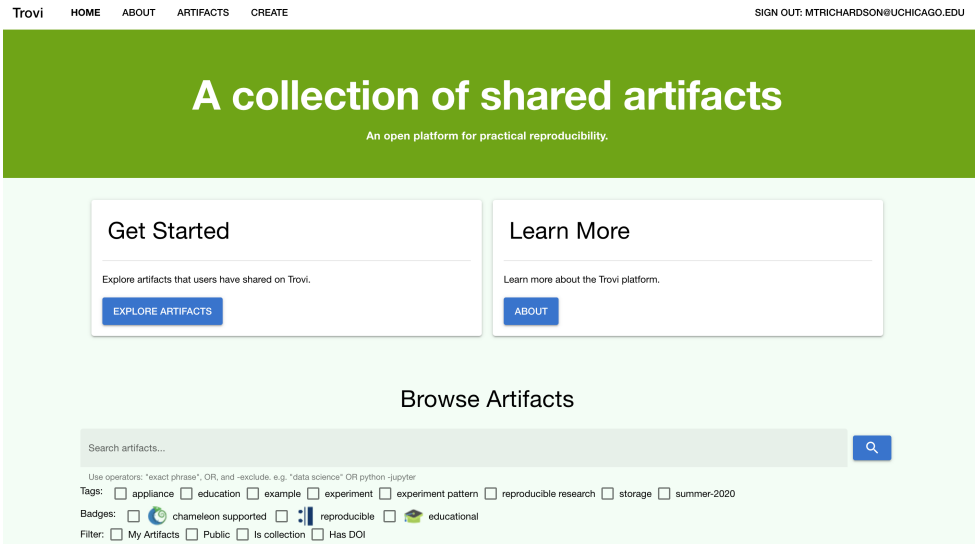
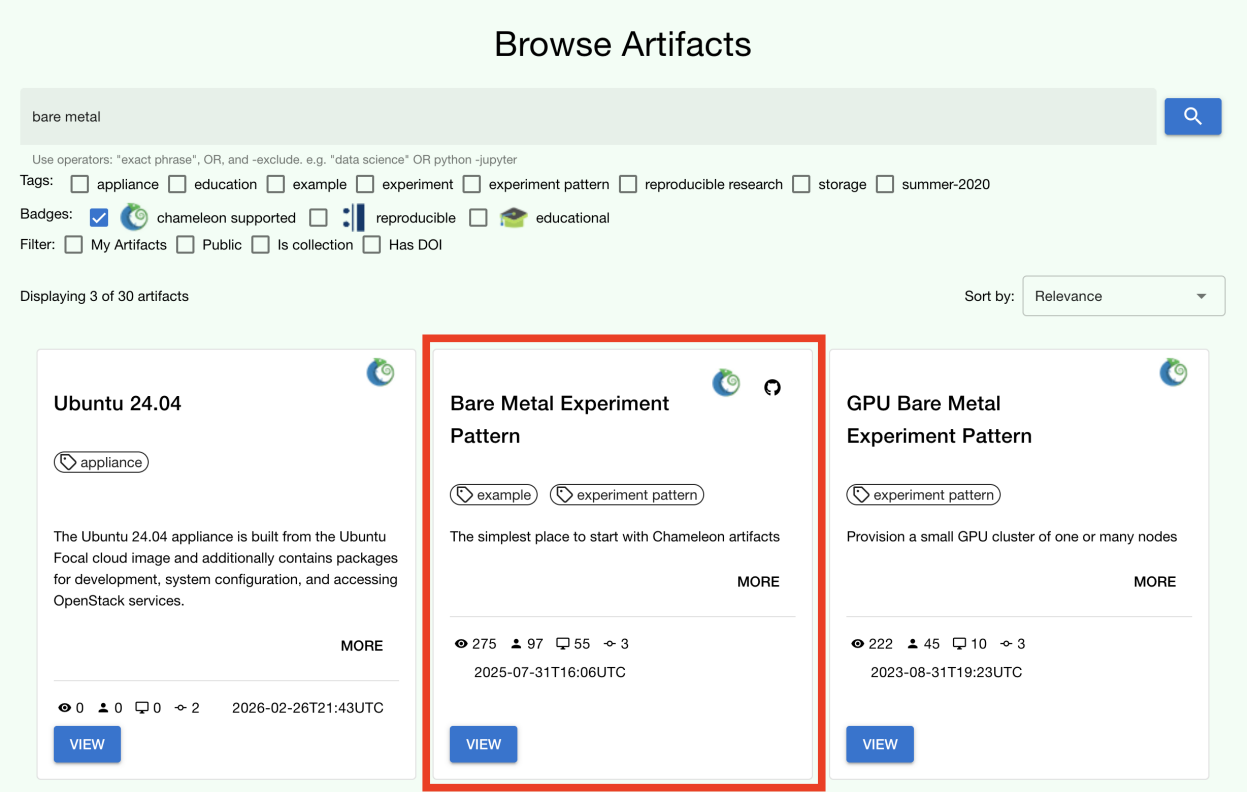


Fig. 1: Jupyter Interface will start a server.



Chameleon offers tutorials and experimental pattern notebooks on Trovi (see collection here). We'll use one now to see how we can accomplish the same basic set up on Chameleon that we achieved in our previous section.

Go to the Trovi repository (after logging in to the site if you aren't already). The artifact we will use today is called the Bare Metal Experiment Pattern. You can type "Bare Metal" in the search bar to filter the results. You can also filter for this artifact by selecting the Chameleon badge icon ( ) on the side bar to view all of the Chameleon-supported artifacts. We can also filter by tag, for example the "experiment pattern" tag.

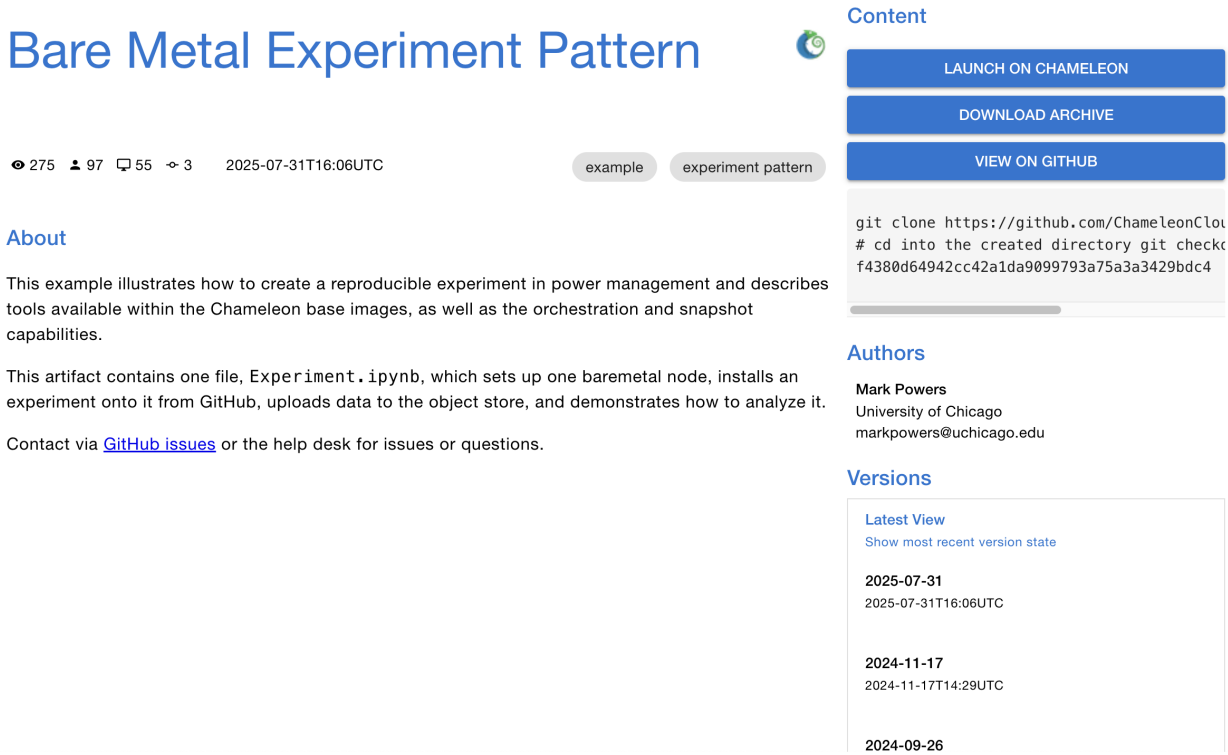


Tip

Want to publish your own experiment on Trovi or import an existing GitHub repository? See our Trovi tips blog post for a **step-by-step walkthrough of the full artifact lifecycle, from packaging to publication**.

Be sure to **check out our additional templates** with more advanced topics on Trovi. The best part about these templates is that you can easily reuse the code to start writing your own artifacts.

To launch the artifact, click on the title. On the next page, you will see the following:



Click on the "**Launch on Chameleon**" button to start Jupyter. This loading page should look familiar to the loading page when we launched the Jupyter Interface above.

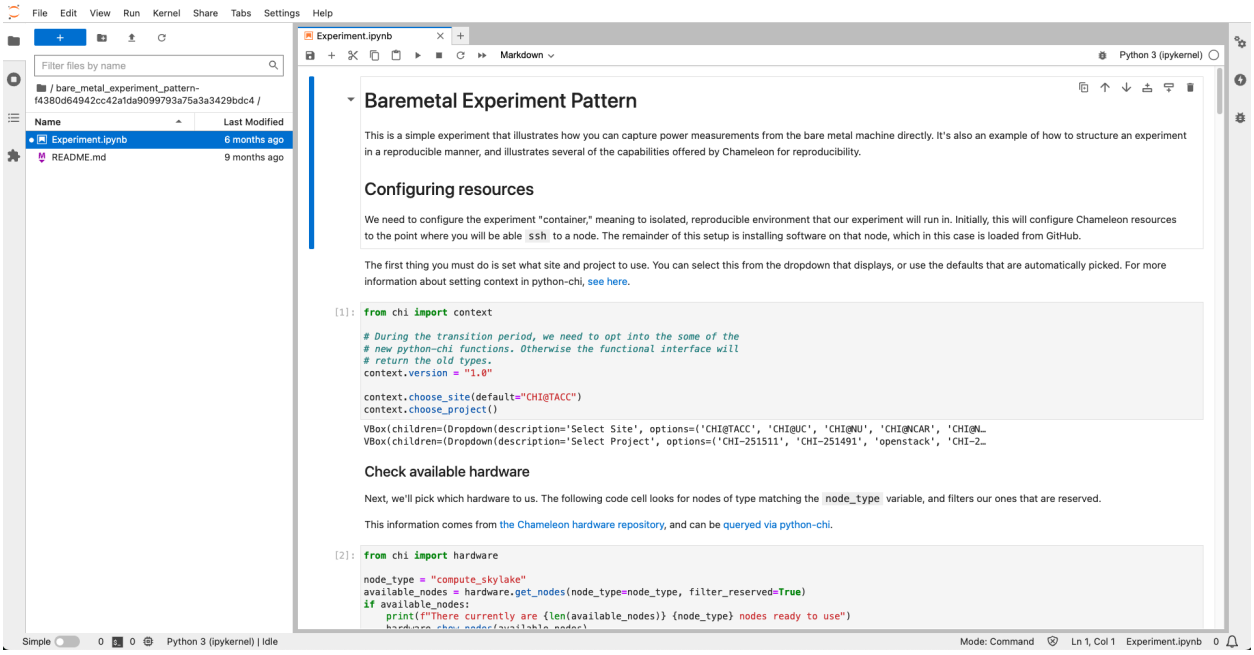
Once Jupyter has loaded, we will have the artifact directory available in our workspace. Your directory should include the following files:

```
$ ls
README.md              Experiment.ipynb            scrips
```

We can click on the directory and open the `README.md` file, which provides some documentation on this artifact, including approximately how long it takes to run and any additional requirements.

Let's now open the `Experiment.ipynb` file.

3.3 Getting started with `python-chi`: bare metal experiment pattern



Jupyter Notebook allows developers to mix text (rendered as Markdown) and code in one file. This mixture of content enhances the experience of running code, because documentation can be provided to clarify the code blocks that run. We can see at the start of the notebook a few blocks of text. If we scroll down to the "Configuration" section, we will see our first block of code. Let's dive in!

Setting the Site and Project

As required when working through the Chameleon GUI, we need to set our active project and pick a testbed site to use before we can continue. This requires a Chameleon account and membership to an active project.

Once we have our project and site, we can use python-chi to set these parameters via the `chi.context` module.

```
import chi

chi.use_site("CHI@UC")
chi.use_project("CHI-XXXXXX")  # Replace with your project name
```

This code imports the python-chi module, calls `use_site` to target a Chameleon site, and `use_project` to set the active project. All subsequent API calls — leases, instances, networks — will be sent to that site and billed to that project. You can call these again at any point to switch context.

> **Tip**
>
> In a Jupyter Notebook, you can use `chi.context.choose_site()` and `chi.context.choose_project()` for interactive dropdown menus instead of hard-coding the values. See the chi.context module docs for the full API, and our python-chi 1.0 blog post for a walkthrough of all the new Jupyter widget features introduced in that release.

Discover Hardware

python-chi now supports hardware discovery via the `chi.hardware` module, mirroring what you can do on the Hardware Discovery web page. This is useful for finding available nodes and checking when specific hardware is free before committing to a reservation.

```
from chi import hardware

# Display an interactive, filterable table of all nodes at the current site
hardware.show_nodes()

# Or filter programmatically - e.g. only GPU nodes with at least 32 CPUs
nodes = hardware.get_nodes(gpu=True, min_number_cpu=32)
```

To check when a specific node is next available, use `next_free_timeslot`:

```
node = hardware.get_nodes(node_type="compute_cascadelake_r")[0]
start, end = node.next_free_timeslot(minimum_hours=3)
print(f"Next free slot: {start} → {end}")
```

See the chi.hardware module docs for the full list of filter options and methods available on `Node` objects.

Create a Reservation

After we set our site and project code, we can now create a lease. The code below uses the `Lease` class to create a reservation for one floating IP and one bare metal host with the node type `compute_cascadelake_r`. Notice that we are setting the same parameters that we had to include in the form we used to create a lease on the GUI.

```
import os
from chi import lease
from datetime import timedelta

l = lease.Lease(
   name=f"{os.getenv('USER')}-power-management",
   duration=timedelta(hours=3)
)
l.add_node_reservation(node_type="compute_cascadelake_r", amount=1)
l.add_fip_reservation(amount=1)
l.submit(wait_for_active=True)
```

See the chi.lease module docs for advanced options, including network reservations, KVM flavor reservations, and setting explicit start/end times for advanced scheduling.

Create an Instance

We can now configure and launch our instance on the node that we reserved.

```
from chi import server

s = server.Server(
   name=f"{os.getenv('USER')}-power-management",
   reservation_id=l.node_reservations[0]["id"],
   image_name="CC-Ubuntu24.04"
)
s.submit(wait_for_active=True)
```

This code uses the `Server` class to spin up an instance. We can specify which image we want to use by referring to its name (in this case `CC-Ubuntu24.04`). (To see the name of an image, you can look it up in the Appliances Catalog on Trovi by filtering for the **appliance** tag.) We also need to provide the reservation ID from our lease, which we can access from the lease's `node_reservations` list. See the chi.server module docs for the full `Server` class API, including `flavor_name`, `network_name`, and `keypair` parameters.

> **Note**
>
> We are *not* specifying a key pair here, because when you use Chameleon through the Jupyter Interface, a key pair is automatically generated in the Jupyter environment and associated with your Chameleon account. By default, the `Server` class will include this key pair in any instance you create from the Jupyter Interface and will use it in other methods that allow you to SSH to the instance. You can specify a different key pair using the `key_name` parameter.

Connecting to and Running Scripts on the Instance

After our server is running (remember, this can take up to 20 minutes in some cases; now is a good time to take a coffee break), we will association our instance with the reserved floating IP and then check our connectivity to the node based using the Server class `check_connectivity` method.

```
floating_ip = l.get_reserved_floating_ips()[0]
s.associate_floating_ip(floating_ip)
s.check_connectivity(host=floating_ip)
```

We have now associated our floating IP and verified our connection to the instance via the floating IP. We can then use `execute` method to upload scripts to our instance for setting up our experiment, running it, and storing the results in Chameleon *object storage*.

```
# Clone git repo with experiment source code
my_server.execute("git clone https://github.com/ChameleonCloud/bare_metal_experiment_
↪pattern")

# Run setup script
my_server.execute("bash bare_metal_experiment_pattern/scripts/setup.sh")
# Run experiment script for N iterations
iterations = 1
for i in range(iterations):
   my_server.execute("bash bare_metal_experiment_pattern/scripts/run_experiment.sh 10")
```

From this point, the remaining code blocks in this notebook will download the data locally from object storage and then plot figures using the experiment data. As an exercise, try to see if you can replicate the experiment in this tutorial on a different node type like a `skylake` or one of our many nodes with a GPU!

Congratulations! You just created your first lease and instance on Chameleon without ever leaving the comforts of your Jupyter Notebook!

> **Note**
>
> This guide covers bare metal instances via the `chi.server` module. python-chi also supports container-based edge computing on CHI@Edge via the chi.container module — see the CHI@Edge docs for details.

Be sure to check out our additional tutorials on Trovi to continue your learning!

CHAPTER

FOUR

CHANGELOG

If you are not redirected automatically, please visit the Chameleon Cloud Changelog.

CHAPTER

FIVE

SIGN IN WITH FEDERATED IDENTITY

Federated login enables users to use a single set of credentials to log into many different services. For example, federated login allows you to use your university or other institutional credentials to log into Chameleon–there is no need to create a new account. In addition, since federated login is supported by many testbeds and services across scientific infrastructures you will be able to sign in once and use multiple services.

Chameleon uses Globus Auth, a popular authentication service, to implement federated login and federates with entities supported by Globus. **We strongly recommend using federated login as it's the simplest and most convenient way to access Chameleon.** Users can sign in using their existing institutional account if their institution is an InCommon member, use their Google account, or create a Globus ID tied to an email and password that they provide.

Note

This page covers browser-based login to the Chameleon user portal, the testbed GUI, and the Jupyter environment. The *Command Line Interface* authenticates separately — signing in here does not by itself give you CLI credentials. See *CLI authentication* to set up CLI access.

5.1 Logging in

To log in to the Chameleon user portal, where you can manage your projects, user profile, and submit Help Desk tickets, use the "Log in" button.

To log in to any of the testbed sites (CHI@TACC, CHI@UC, CHI@NCAR, CHI@Edge, KVM@TACC) or the *Jupyter environment*, just click their item in the "Experiment" dropdown on chameleoncloud.org. The login process is triggered automatically.

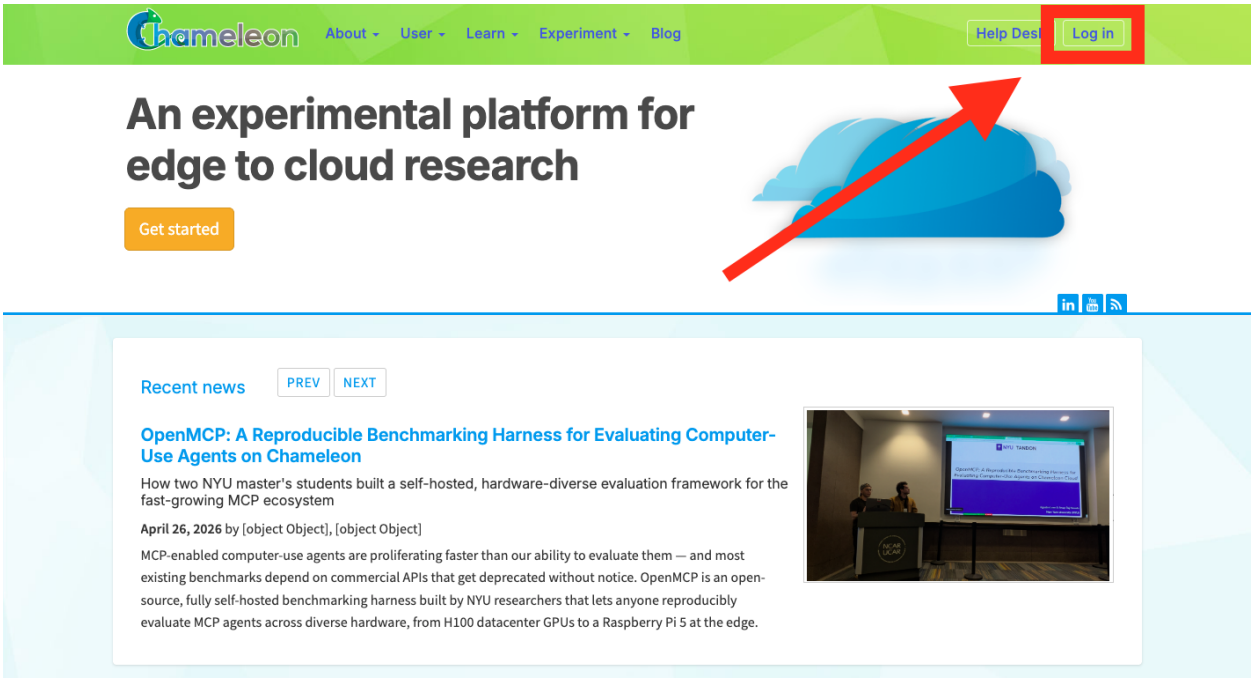
Warning

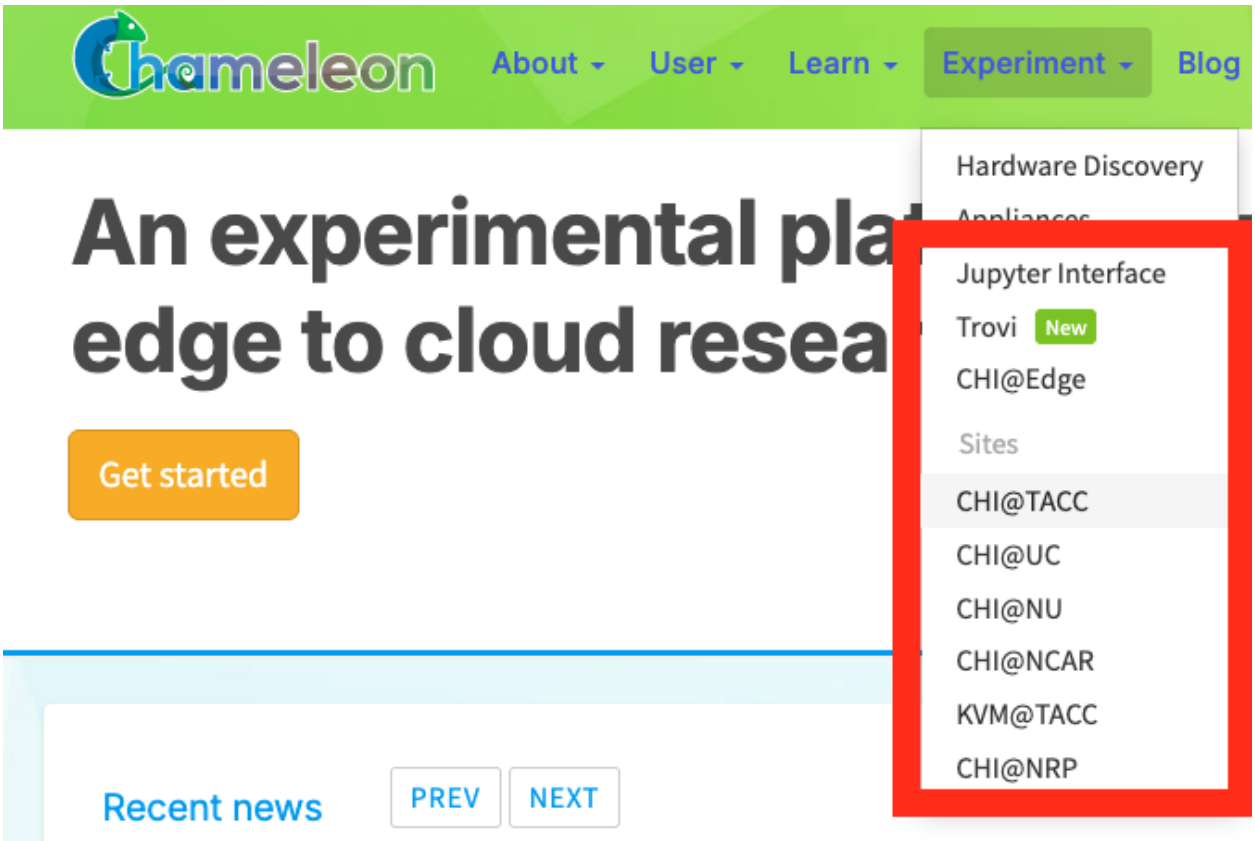
You must be part of a project with an active allocation to use the CHI sites! Refer to our *project management guide* for details on creating a project or joining an existing one.

Tip

You can bookmark the URLs to the testbed sites and Jupyter environment if you want to access them directly in the future.

You will be taken to a Single Sign On (SSO) page with several authentication options.


About
User
Learn
Experiment
Blog
Log in
An experimental platform for edge to cloud research
Get started
Recent news
PREV
NEXT
OpenMCP: A Reproducible Benchmarking Harness for Evaluating Computer-Use Agents on Chameleon
How two NYU master's students built a self-hosted, hardware-diverse evaluation framework for the fast-growing MCP ecosystem
April 26, 2026 by [object Object], [object Object]
MCP-enabled computer-use agents are proliferating faster than our ability to evaluate them — and most existing benchmarks depend on commercial APIs that get deprecated without notice. OpenMCP is an open-source, fully self-hosted benchmarking harness built by NYU researchers that lets anyone reproducibly evaluate MCP agents across diverse hardware, from H100 datacenter GPUs to a Raspberry Pi 5 at the edge.


About
User
Learn
Experiment
Blog
Hardware Discovery
Jupyter Interface
Trovi
New
CHI@Edge
Sites
CHI@TACC
CHI@UC
CHI@NU
CHI@NCAR
KVM@TACC
CHI@NRP
Get started
Recent news
PREV
NEXT

5.1.1 Authentication options

- **Sign in via federated identity RECOMMENDED**: This is the simplest way to access Chameleon! Use your existing institution, research lab, or university credentials to log in. This requires your host institution to participate in the InCommon federation. Most major universities and research institutions are already members.
- **Google**: Sign in with any Google account if your institution isn't part of InCommon or if you prefer using your Google credentials.
- **ORCiD**: Sign in with a valid ORCiD account for research credential integration.

Warning

You may not find your institution on Globus. If so, you can still create an account with another identity, such as Google or ORCiD. However, we encourage users to sign up using their institution, as it helps the Chameleon operators verify user identity, which is an essential step to getting Principal Investigator status on the testbed.

Note

Chameleon has fully migrated to federated identity authentication. Legacy Chameleon username/password accounts are no longer supported.

5.2 Terms and conditions

When creating an account, you will be asked to review and consent to the Acceptable Use Policy, which governs all use of Chameleon resources and applies to every project on the testbed.

As part of these terms, among others, you are requested to acknowledge Chameleon in publications produced using the testbed. See our FAQ for information on how to cite Chameleon and the suggested acknowledgement text. The FAQ also covers other common policy questions about allocation charges, usage best practices, and more.

5.3 Account linking

Chameleon accounts are linked to a federated identity provider, Globus. You'll need to (re-)link your account if you're no longer able to log in via your original account source (e.g. you move institutions), or if you're stuck in an "account already exists" login loop because you have two separate Globus identities (e.g. a Google sign-in and an institutional login) pointing at two different Chameleon accounts. Either way, the fix below assumes you have access to the same email address as your existing account.

Note

Previously Chameleon also supported federated identity via TACC, but this is no longer supported after January 2026. If you previously were using TACC to login to Chameleon, we recommend following the steps below to link your account to Globus.

1. Log out of the Chameleon site via the menu on the top right of the web page on whatever platform you are using.
2. Log out of Chameleon's authentication service.
3. Sign into Globus with the same primary email address as your Chameleon account. If your institution is not supported by Globus login, we recommend creating a Globus ID or ORCID account.

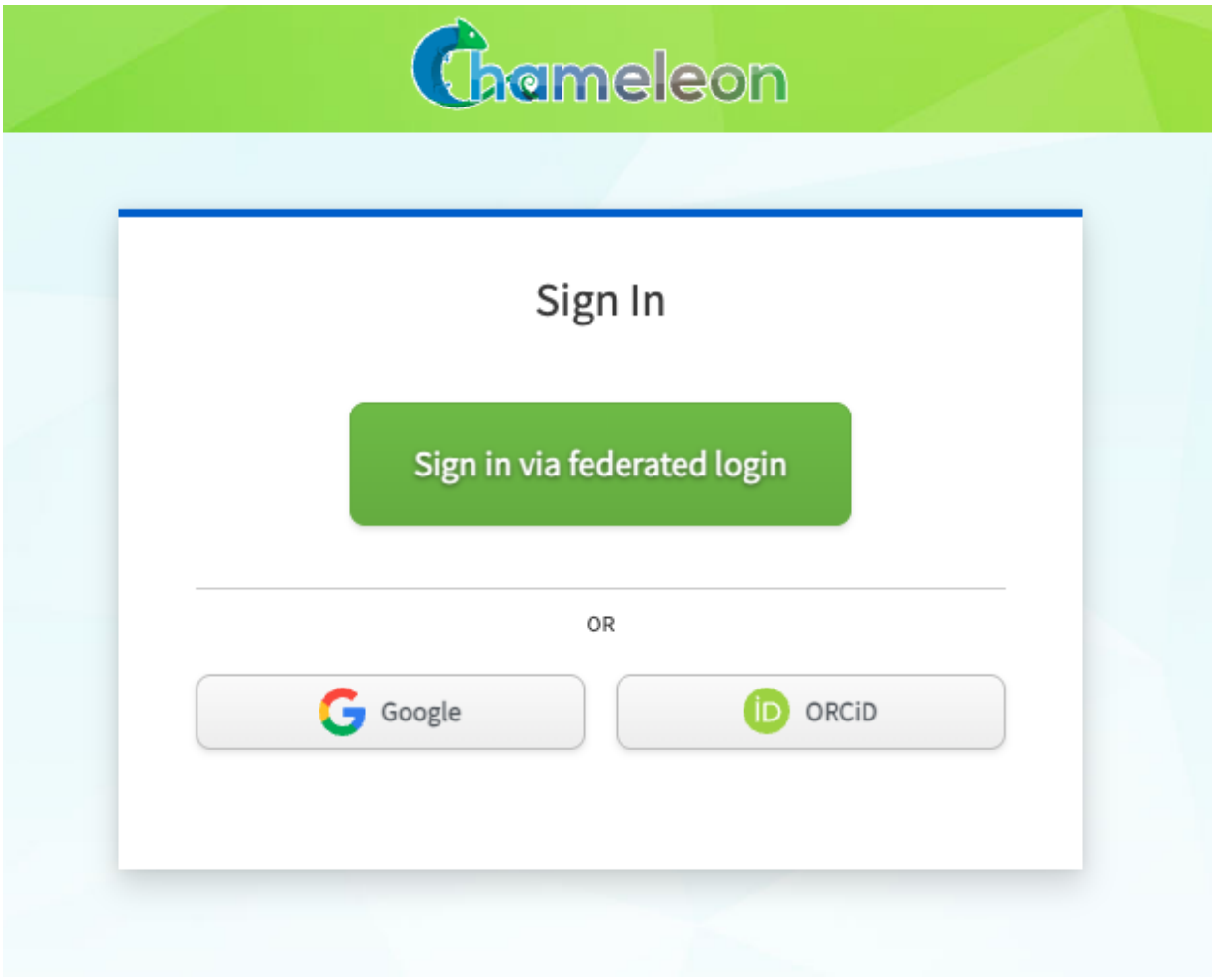


Fig. 1: The Single Sign On (SSO) portal login page.

4. Return to the Chameleon website from step 1, and click "Log in" and then "Sign in via federated login".
5. If an existing Chameleon account was found with the same email address, you will see a message saying "You need to verify your email address to link your account with Globus Auth." Check your email inbox for a verification link. Once you click this link, your Chameleon account will now be linked to the Globus account from step 3, and you should use this login method when using Chameleon in the future.

If you don't see this message, then your Globus account is using an email address that Chameleon doesn't have registered. Please contact the Help Desk for further assistance regarding this or any other issue with the account linking process.

5.3.1 Lost access to your original email

If you no longer have access to the email address on your existing account — for example, you graduated, changed jobs, or your institution migrated email domains — you won't be able to complete the verification step above, since there's no inbox to receive the link.

In this case, contact the Help Desk directly and include:

- The old email address on your existing Chameleon account
- The new email address you'd like to use going forward
- Your institution (old and new, if it changed)

Staff will use this information to verify your identity and relink your account manually.

5.4 Troubleshooting login issues

If you experience difficulty logging in, try these solutions:

Common Authentication Issues:

- **"An account with this email already exists" / repeated login loops**: This almost always means you have two separate Globus identities (for example, a Google sign-in and an institutional SSO login) pointing at two different Chameleon accounts. Follow the *Account linking* steps above to link them into one. If two accounts still won't merge afterward, contact our *Help Desk* and ask staff to merge them manually.
- **Prompted for a username and password**: Federated accounts never have a separate Chameleon password to reset — there's nothing behind that screen. Landing here instead of your institution's login page is a symptom of the same unlinked-identity issue above, not a forgotten credential; resetting a password will not fix it.
- **"Account is disabled, contact your administrator"**: This means your account was automatically locked out after repeated failed login or authentication attempts (often triggered by scripted or CLI logins using stale credentials) — it is not a ban, and it isn't something you can resolve yourself. Contact our *Help Desk* and staff will re-enable it for you.
- **Institutional credentials not working**: Ensure your institutional credentials are up-to-date and correctly linked to your Chameleon account
- **Browser issues**: Clear your browser cache and cookies, then try logging in again

Getting Help:

For persistent login issues, contact our *Help Desk* with details about:

- Which authentication method you're trying to use
- Any error messages you're seeing
- Your institutional affiliation (if using federated login)

CHAPTER

SIX

PI ELIGIBILITY

6.1 Overview

> **Warning**
>
> **Undergraduate and Graduate Students**: Please note that if you are a student (even a postdoc or PhD), you are **not eligible** to request PI status. **Ask your faculty advisor or research supervisor to request PI status and add you to their project**. This is best approach for student researchers who want to work with Chameleon resources. See the blog post Getting Access to Chameleon as a Student Researcher for a walkthrough of this and other access options, such as Daypass.

Chameleon PIs carry significant responsibility for the users on their projects; we therefore limit PI eligibility to individuals from the following groups:

- **Academic institutions**: This eligibility criterion covers research scientists, research staff, and faculty members in supervisory positions at academic institutions. Graduate and PhD student researchers (including those serving as paid research assistants) are **not** typically considered eligible for PI status on Chameleon. Students should instead ask their faculty advisor to request PI status and give them access to a project.
- **Federal agencies such as national labs, R&D centers, and institutes**: Research staff employed by federal agencies or non-NSF Federally Funded R&D Centers (FFRDCs) are eligible to apply for an allocation.
- **International research institutions**: To promote intellectual exchange and federation with institutions abroad we support a limited number of international PIs with ongoing, active collaborations with scientists in the US.
- **NSF Graduate Student Fellows**: While in most cases, a graduate student is ineligible to be PI of an allocation request, an exception is made for NSF Graduate Student Fellows. Recipients of these NSF awards can submit requests for Startup allocations as long as they include supporting documentation (grant number or an award letter) as part of the request submission.
- **Independent museums, observatories, libraries, research laboratories, professional societies, and similar organizations** in the United States that are directly associated with educational or research activities are eligible.
- **State educational offices or organizations and local school districts** may submit allocation requests intended to broaden the impact, accelerate the pace, and increase the effectiveness of improvements in science, mathematics, and engineering education in both K-12 and post-secondary levels. A teacher or educator at an accredited public or private K-12 school is eligible to apply for an allocation as PI.

We do occasionally provide case-by-case exceptions to this guideline in well-justified cases.

6.2 How to request PI eligibility

If you meet the eligibility criteria above, you can request PI status on Chameleon by following these steps:

1. **Complete your user profile**: Ensure your Chameleon profile is complete with your institutional affiliation and contact information.
2. **Request PI eligibility**: In your profile page, click the "Edit Profile" button, then check the "Request PI Eligibility" checkbox and save your profile.
3. **Wait for review**: Chameleon PI status requests are typically reviewed and approved within one business day.
4. **Create your first project**: Once approved, you can create a new project by going to the Projects Dashboard and clicking "Create a Project."

For questions about PI eligibility or if you believe you have a special case that warrants consideration, contact our *Help Desk*.

CHAPTER

SEVEN

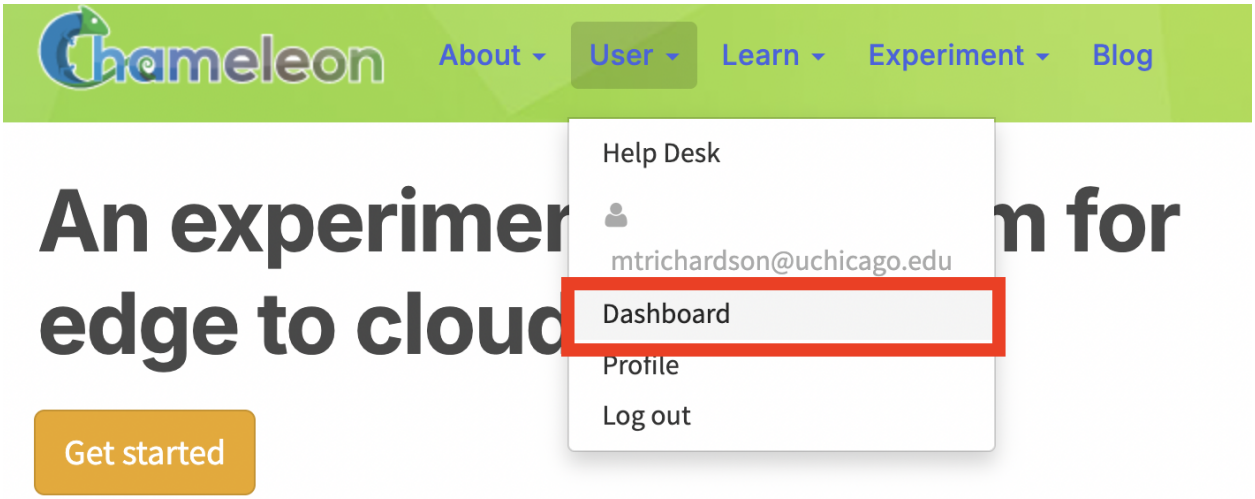
PROJECT MANAGEMENT

7.1 Overview

Projects are the fundamental organizing unit for research on Chameleon. Each project provides:

- **Resource allocations** measured in Service Units (SUs) for computational time
- **User management** with role-based access control (PI, Manager, Member)
- **Resource isolation** with dedicated security groups, networks, and storage
- **Usage tracking** and billing management for fair resource sharing
- **Publication tracking** to record research outputs associated with the project

All project management tasks are performed through the Chameleon portal. Log in to the portal and go to your User Dashboard. You can access your *Dashboard* via the dropdown list on the top right of the screen.



Quick Navigation

- *Dashboard overview* - View projects, tickets, and outages
- *Creating a project* - Start a new research or education project

- *Managing users* - Add/remove team members and set roles
- *Allocation management* - Request renewals and recharges
- *Publications tracking* - Maintain research output records

Related Documentation

- *PI eligibility requirements* - Criteria for creating projects
- *User authentication* - Login and account configuration
- *Getting help* - Support channels and community resources

7.2 Dashboard

The Dashboard's **Overview** page consists of thre control panels - the *Active Projects* control panel, the *Ongoing Outages* panel, and the *Open Tickets* panel.

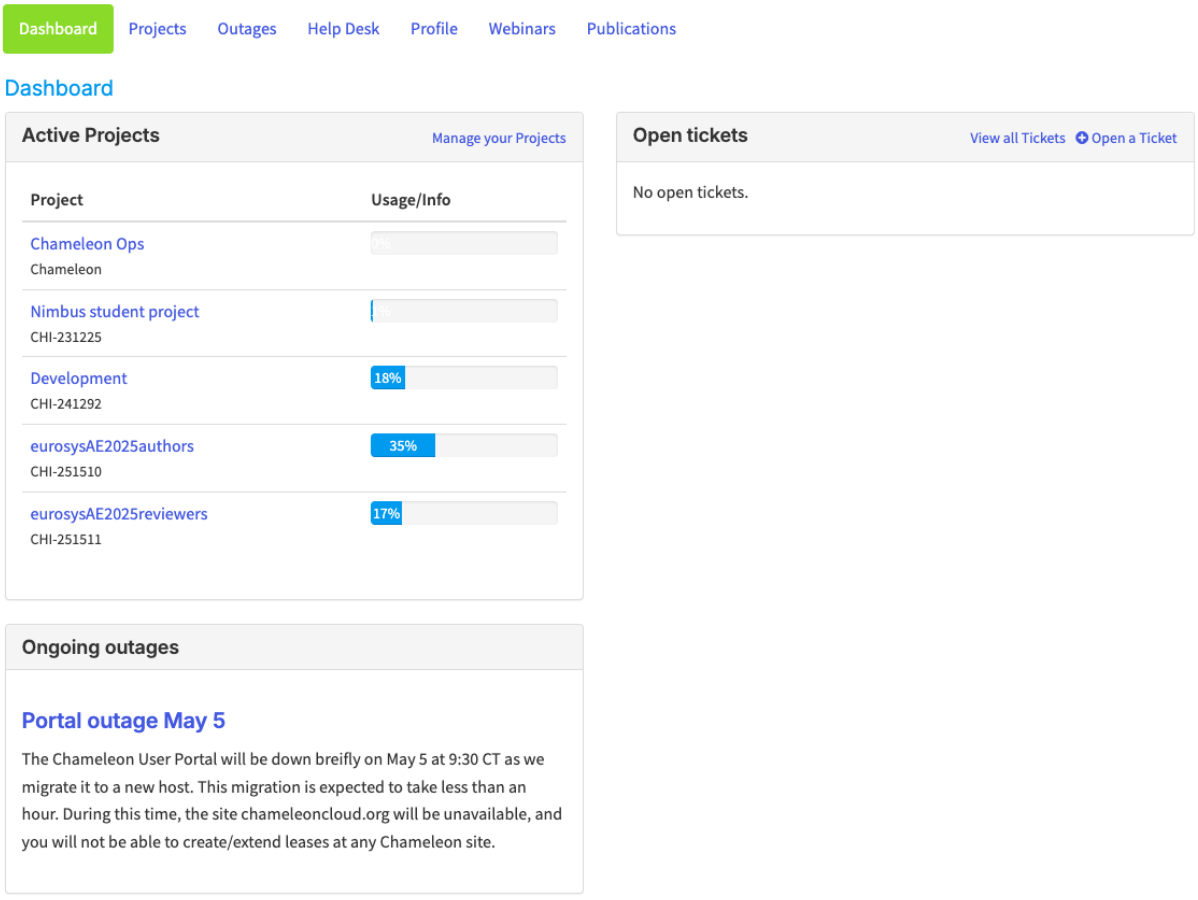


Fig. 1: The project dashboard

The *Active Projects* control panel allows you to view all your active projects and their current usage. You may click on a project to view details.

The *Open Tickets* panel lists all your active help desk tickets. In addition, you can Open a Ticket via the *Open Tickets* panel.

The *Ongoing Outages* panel displays information about ongoing outages on the system. Learn more about *Outages*.

7.3 Projects

The Dashboard's Projects Page shows a list of all your active and inactive projects.

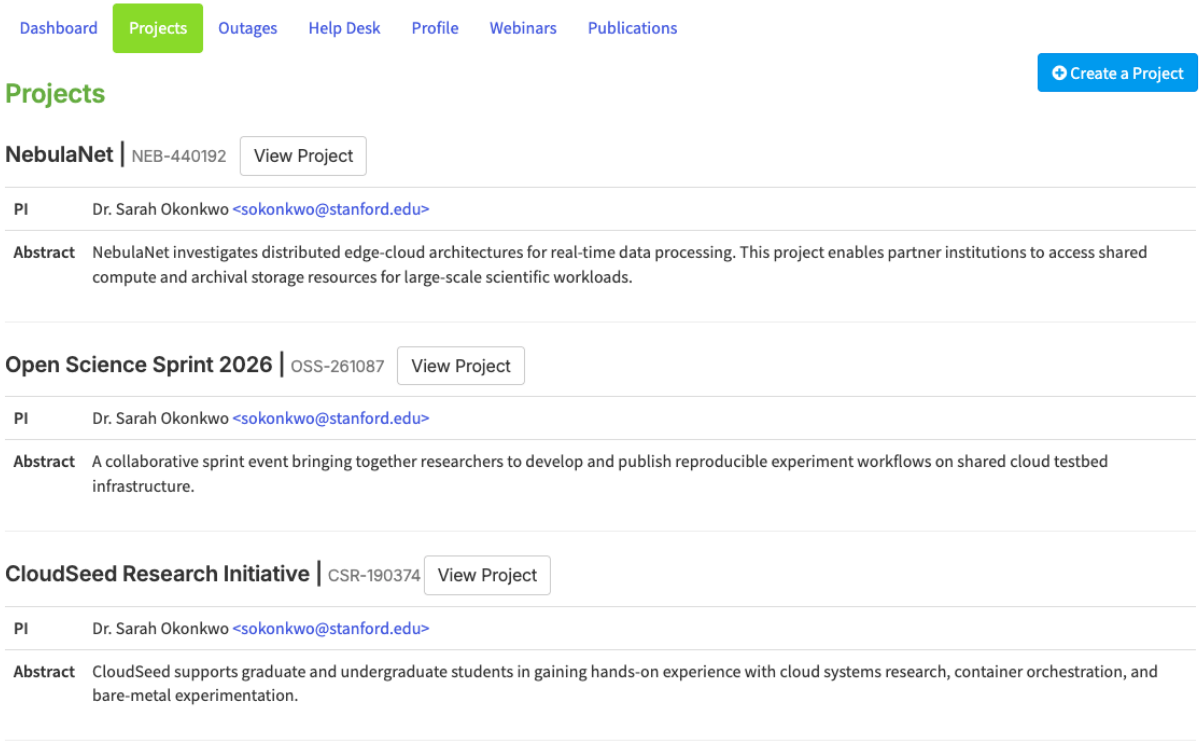


Fig. 2: Project list

Each individual *Project* has its own:

- Service Unit allocations
- Users that have access to the project
- System resources such as *Security Groups*, *Floating IP Addresses* and *Instances*
- Assets such as snapshots, object containers, metrics and network configuration

7.3.1 Creating a project

To create a project, click the *+Create a Project* button. After filling out and submitting the request form, a system administrator will review your request and notify you once your project gets approved. Project durations are six months with a default allocation of 20,000 *Service units*.

Service units

One Service Unit (SU) is equivalent to one hour of wall clock time on a base bare metal server. For full details on SU rates for specialized hardware and KVM instances, see our FAQ.

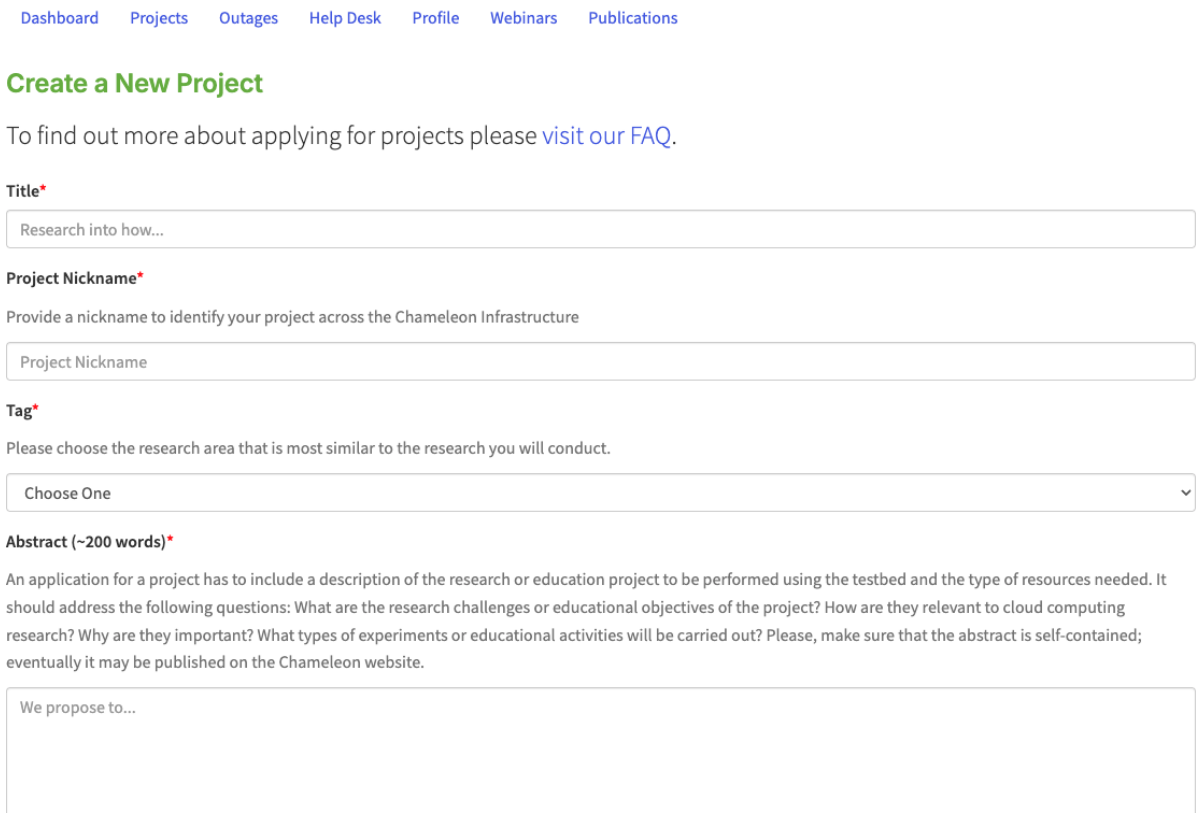


Fig. 3: The Create a New Project form

Project details

Clicking on a project from either the *Dashboard* main page or the *Projects* page will allow you to manage one of your approved *Projects*.

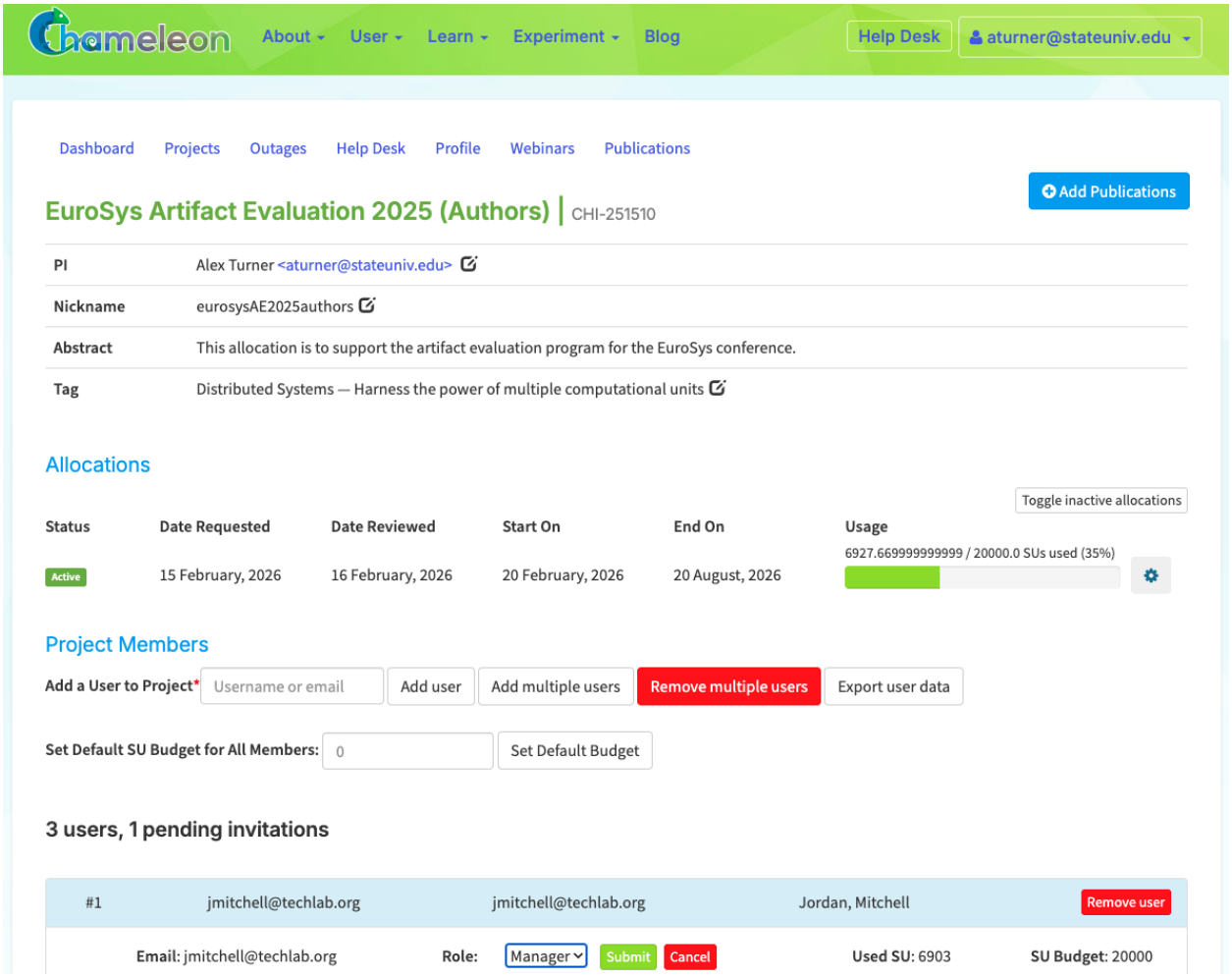


Fig. 4: Project details

In the details page of your project, you may *recharge or extend your allocation*, *view allocation usage details*, and *manage users* of your project.

7.3.2 Recharge or extend your allocation

In the *Allocations* section of your *Project details*, you may view your project start and end dates, current *Service Unit* usage and request a *Service Unit* recharge or project extension. To request a *Service Unit* recharge or *Project* extension, click the *gear* button at the end of the allocation row, and then click *Recharge/Extend Allocation*. When requesting renewal or recharge of the allocations, we may ask you to update your *publications dashboard*, so keeping it up to date now can save you time later! For questions about allocation management, visit our Community Forum or contact the *Help Desk*.

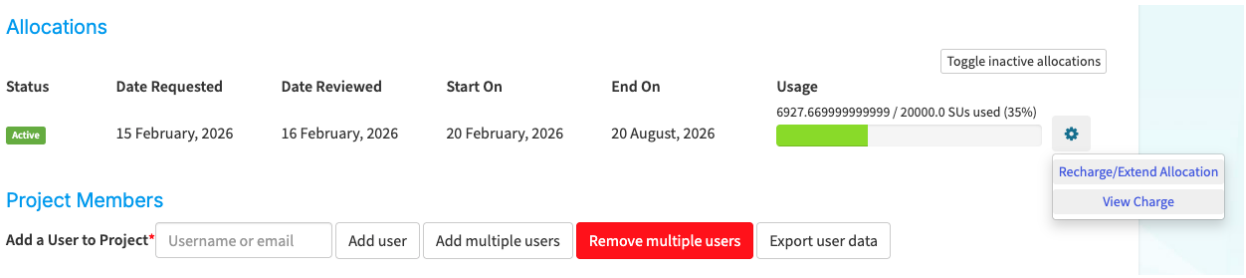


Fig. 5: Allocation actions

7.3.3 View allocation usage details

Region	Resource ID	Resource Type	Start On	End On	Used SUs
CHI@TACC	33aa7e36-755f-4cf2-a04a-731191e1449d	physical:host	Feb. 25, 2026, 10:34 a.m.	March 5, 2026, 11:33 p.m.	614.93
CHI@TACC	abf5726c-2dc0-4e40-97a1-97a05850058b	virtual:floatingip	Feb. 25, 2026, 10:34 a.m.	March 5, 2026, 11:33 p.m.	0.0
CHI@TACC	d28e9a7a-42ee-4815-89de-9273d1939b1e	virtual:floatingip	March 9, 2026, 3:36 p.m.	March 9, 2026, 3:38 p.m.	0.0
CHI@TACC	1d11a9fa-74f3-46d6-a3d0-35aea2b01986	virtual:floatingip	March 9, 2026, 3:39 p.m.	March 9, 2026, 3:41 p.m.	0.0
CHI@TACC	d64834d1-aac1-4336-9582-6b0b240f4238	virtual:floatingip	March 9, 2026, 3:43 p.m.	March 9, 2026, 3:43 p.m.	0.0
CHI@TACC	c12310af-87b6-4b89-9b74-1ebad6328c27	physical:host	March 9, 2026, 3:45 p.m.	March 9, 2026, 3:49 p.m.	0.21
CHI@TACC	c7be7fd5-f012-4e77-9c70-fcc46e45c5f5	virtual:floatingip	March 9, 2026, 3:50 p.m.	March 9, 2026, 3:50 p.m.	0.0
CHI@TACC	aa9952a7-f3df-41b6-86c5-0798f958335d	physical:host	March 9, 2026, 3:56 p.m.	March 9, 2026, 4:10 p.m.	0.74
CHI@TACC	cec8c3ab-3e4b-4959-835d-ad7cf6e91509	virtual:floatingip	March 9, 2026, 3:56 p.m.	March 9, 2026, 4:10 p.m.	0.0
CHI@UC	5763ba71-e014-41df-99de-0f1d85504970	physical:host	March 9, 2026, 4:03 p.m.	March 14, 2026, 7:31 p.m.	370.4
CHI@UC	c7fd6515-e009-491e-b757-633905f1deaa	virtual:floatingip	March 9, 2026, 4:03 p.m.	March 14, 2026, 7:31 p.m.	0.0
CHI@TACC	58da7d12-df2e-4970-ad78-50160fd1330f	physical:host	March 9, 2026, 4:12 p.m.	March 9, 2026, 4:13 p.m.	0.06
CHI@TACC	e3233013-3e00-4fd1-aab0-1f31b63c3972	virtual:floatingip	March 9, 2026, 4:12 p.m.	March 9, 2026, 4:13 p.m.	0.0
CHI@TACC	5cbbc6d4-0f4b-4999-be9f-3206edda7b15	virtual:floatingip	March 9, 2026, 4:14 p.m.	March 9, 2026, 4:14 p.m.	0.0
CHI@TACC	45e1af1f-599d-4190-b681-b1508a840522	network	March 9, 2026, 4:14 p.m.	March 9, 2026, 4:14 p.m.	0.01
CHI@TACC	cf3f9908-b272-4dea-8dd4-dd3033fdc51b	virtual:floatingip	March 9, 2026, 4:15 p.m.	March 10, 2026, 4:14 p.m.	0.0
CHI@TACC	7d11702f-d90a-4aae-9c68-572f50f12986	physical:host	March 10, 2026, 8:33 a.m.	March 11, 2026, 8:32 a.m.	23.98
CHI@TACC	102110bb-9201-4421-bcdd-641af6c304bd	virtual:floatingip	March 10, 2026, 8:33 a.m.	March 11, 2026, 8:32 a.m.	0.0
CHI@UC	5763ba71-e014-41df-99de-0f1d85504970	physical:host	March 14, 2026, 7:31 p.m.	March 22, 2026, 9:17 a.m.	545.34
CHI@UC	c7fd6515-e009-491e-b757-633905f1deaa	virtual:floatingip	March 14, 2026, 7:31 p.m.	March 22, 2026, 9:17 a.m.	0.0
CHI@TACC	a3b41b3d-4f8e-47ac-a1bf-362bf1b15457	[redacted]	026, 4:08 p.m.	March 19, 2026, 4:08 p.m.	0.02
CHI@TACC	a3b41b3d-4f8e-47ac-a1bf-362bf1b15457	[redacted]	026, 4:08 p.m.	March 27, 2026, 4:07 p.m.	575.93

Fig. 6: Allocation usage details

To view the allocation usage details, in the *Allocations* section, click the *gear* button at the end of the allocation row, and then click *View Charge*. This will open a modal displaying a list of all charges against your allocation, including who initiated the charge, how many *Service Units* were charged, and what type of charge it was.

7.3.4 Manage users

In the dashboard, you can add or remove users (or "members") from your projects, manage your project members' user roles, and allocate how many SUs each project member can consume on your project allocation.

User roles

To manage user roles for a *Project*, scroll down to the *Project Members* section in the *Project details* page of your dashboard. The table below shows the types of roles that members can have and their privileges.

Role	Description
PI	Each project has only one PI. PI can manage roles of the project members.
Manager	Each project can have multiple Managers. Managers can manage project membership and renew allocations of the project.
Member	Members can only view the list of the project members.

To change the role of a project member, choose a new role from the dropdown and click the *Submit* button to apply the new role to the user, or use the *Cancel* button to cancel the action.

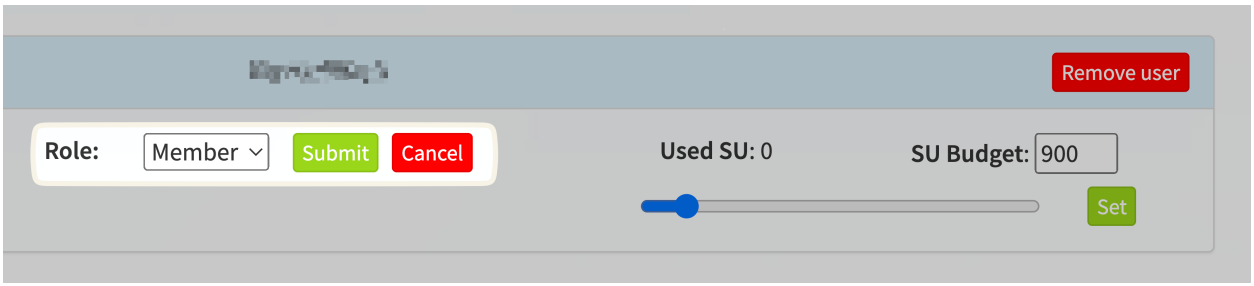


Fig. 7: Manage Role of a User

Set SU budgets for project members

The PI or project managers can set a service unit (SU) budget for each project member. This budget limits the resources that a project member can utilize from the project's total SU allocation. Managing these budgets ensures fair distribution of resources and effective utilization of project resources across multiple project members. Setting a user-specific budget can help when managing resources for a project with lots of members (a large collaborative research project or a classroom project, for instance) and ensure that project allocations are shared effectively between project members.

Project managers (including PIs) can set a default SU budget that is applied to all project members except managers. All new users added to the project will receive the same default SU budget upon joining.

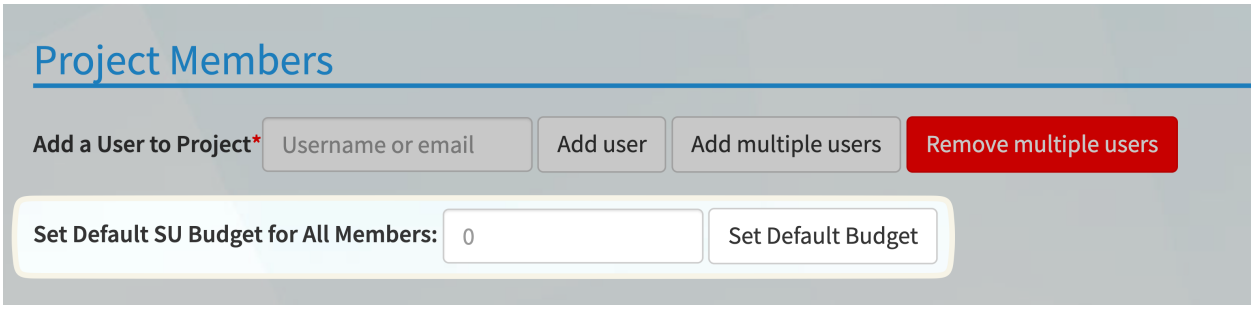


Fig. 8: Set project default budget

Viewing User SU Budgets - Project members will have their SU budget displayed next to their name in the *Project Members* table. This represents the allocation of resources that they can utilize within the project.

Project managers may also set different limits for different users. To adjust the SU budget for a specific user, use the slider or the SU Budget field to *Set* the new budget for an individual user.

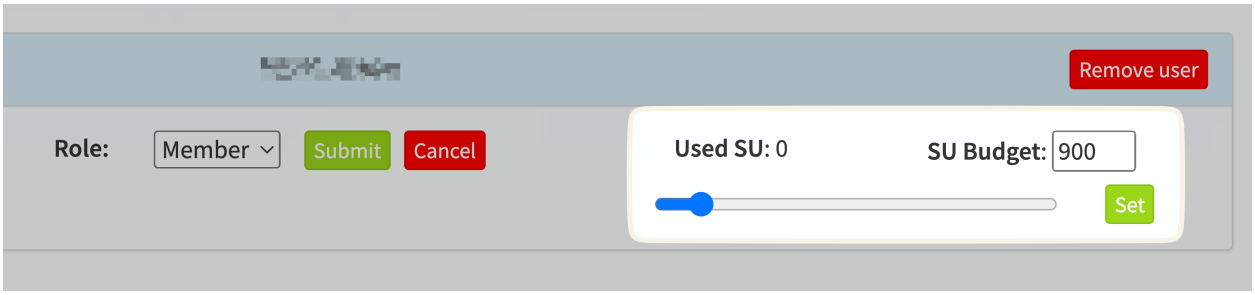


Fig. 9: Adjust SU budget for user

Adding and removing new members

To add or remove users of a *Project*, use the *Project Members* section in the *Project details* page on your dashboard.

You may add a user to your project by filling out their username or email address and clicking the *Add user* button. While each user has their own Chameleon User account independent of your project, they may be added to one or more projects. Being a user of a *Project* **does not** require PI eligibility (see our *PI eligibility guide* for details on project creation requirements).

You may remove a user from your project by locating the user in the user list; clicking the *gear* button at the end of the row; and clicking *Remove user*.

It is also possible to bulk-add a large list of users by clicking the "Add multiple users" button, or remove all users without the Manager role by clicking the "Remove multiple users" button. Additionally, under this option there is a link that you can send to users that will allow them to request to join your project after they sign in to Chameleon. Once a request is made, the managers of a project will be notified, and will need to approve the request.

If there is no user associated with an email address, an invitation will be sent with a link. When someone clicks on this link, they will be prompted to sign in or create an account, and are then automatically added to the project. Invitations show up at the bottom of the members list, and can be deleted or resent if needed. After an invitation is accepted, the user will show up under the *Project Members* section.

7.3.5 Manage publications

To add publications to a project, click the *Add Publications* button in the *Project details* page. Enter the publications in BibTeX format. All regular BibTeX publication types are supported. If you can provide a link, enter it as *note* or *howpublished* using the url package.

To manage the publications you have entered, use the *Publications Dashboard*.

In the dashboard, you may remove a publication from a project by clicking the - button next to the publication text.

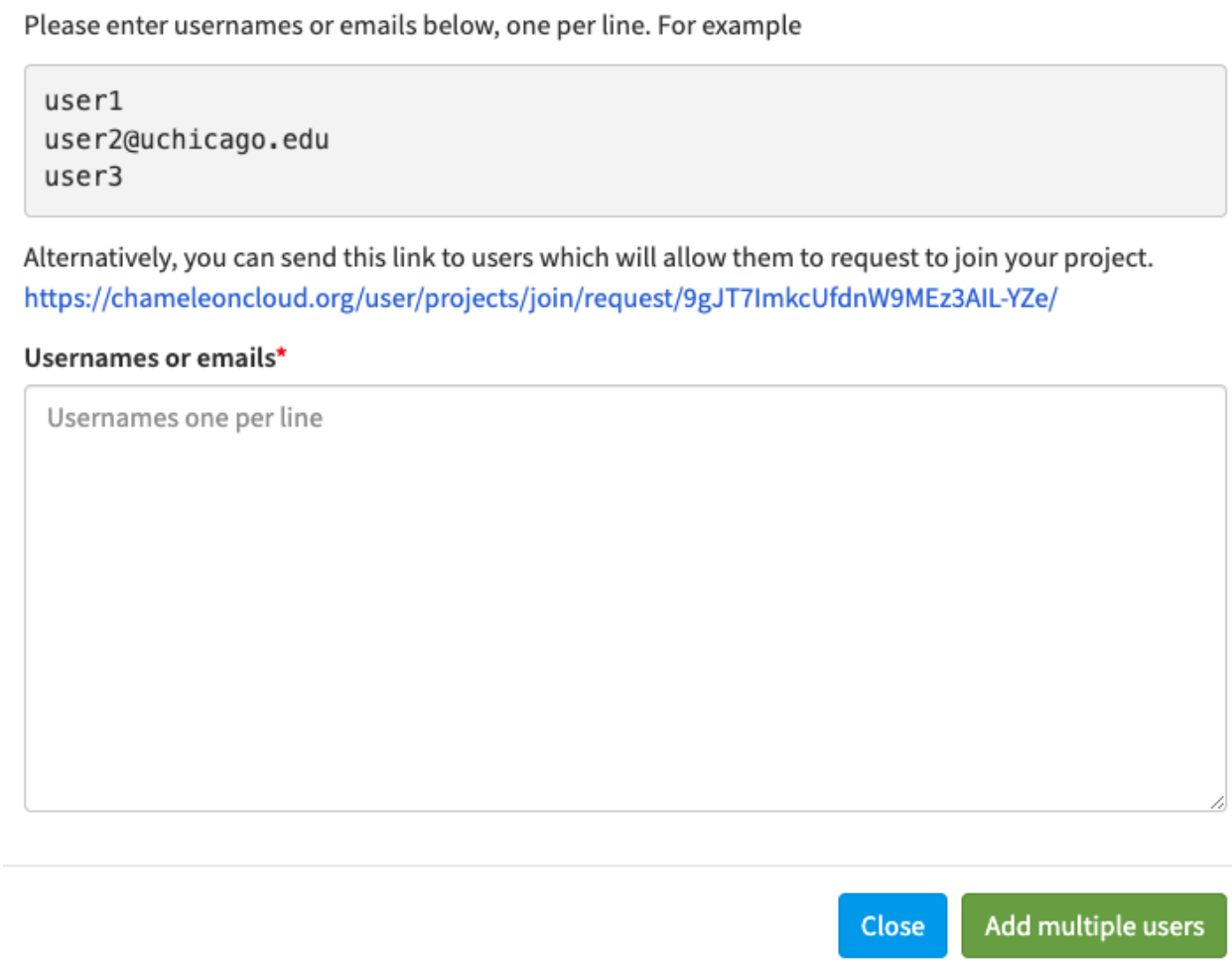


Fig. 10: Adding multiple users

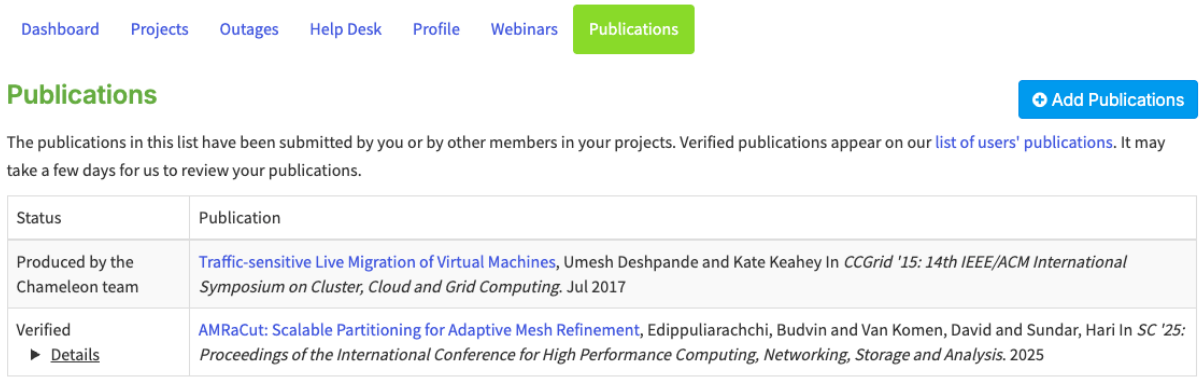


Fig. 11: Publications Dashboard

CHAPTER

EIGHT

USER PROFILE

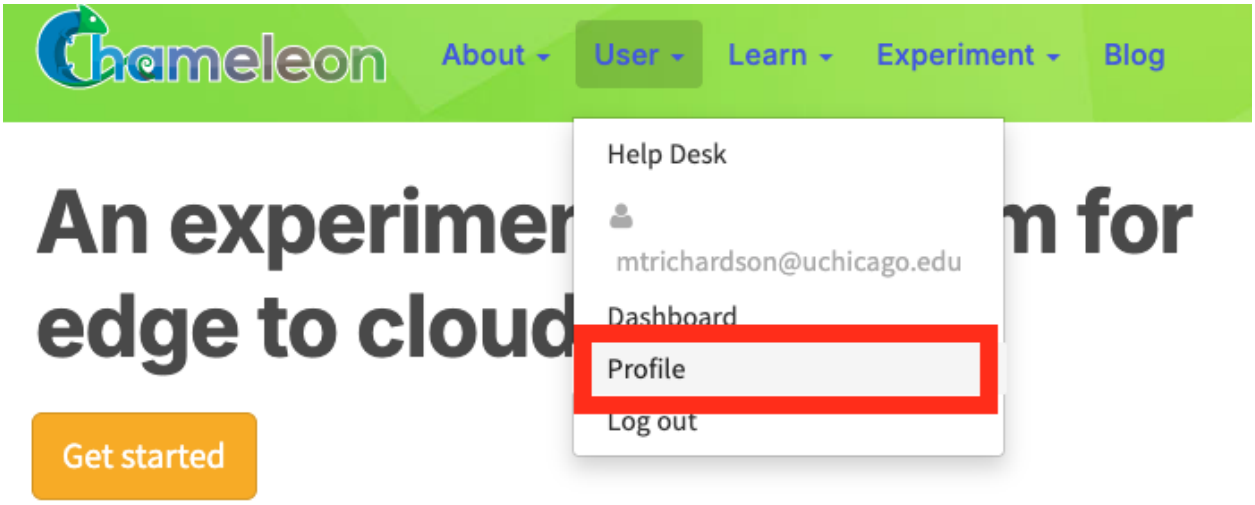


Fig. 1: Navigate to the user profile page through the **User** tab

8.1 Accessing your profile

You can reach your Profile page via:

- **The user dashboard**: Select the **Profile** tab from the *User Dashboard*.
- **The navigation bar**: Click your name in the top-right corner of any portal page, or hover over the **User** tab, and select **Profile** from the dropdown menu.

8.2 Profile overview

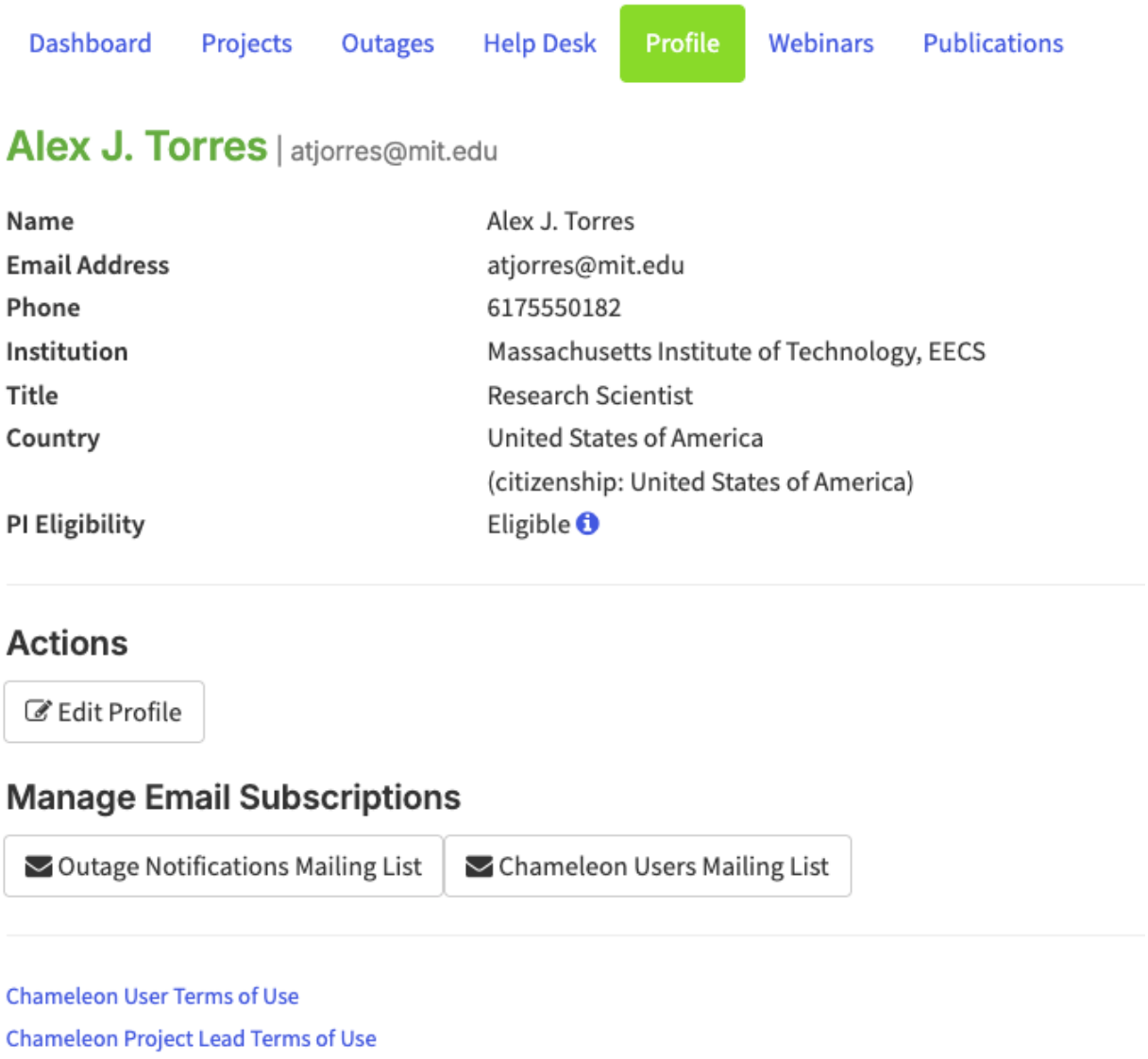


Fig. 2: The Profile page

The Profile page shows summary of your account information:

Field	Description
Name	Your full name
Email Address	The email address associated with your account
Phone	Your contact phone number
Institution	Your institution and department
Title	Your position or role
Country	Your country of residence, with country of citizenship in parentheses
PI Eligibility	Your current PI eligibility status

8.3 Editing your profile

Click **Edit Profile** to open the edit form. Update account fields, then click **Save Profile** to apply your changes or **Cancel** to return to the profile view without saving. You can also **request Principal Investigator (PI) eligibility** if you do not have it already from this page. See more about requesting PI status in *PI Eligibility*.

> **Warning**
>
> **Country of citizenship and name** cannot be edited directly. To these fields, you must file a support ticket.

8.4 Managing email subscriptions

Your profile page provides access to two mailing lists. Click either button to open the external subscription page where you can subscribe or unsubscribe:

- **Outage Notifications Mailing List** — Notifications about system outages
- **Chameleon Users Mailing List** — General Chameleon user community announcements

8.5 Terms of use

Our terms of use documents are available at the bottom of the profile page:

- Chameleon User Terms of Use — Applicable to all users
- Chameleon Project Lead Terms of Use — Applicable to users who act as project leads (PIs); see the "Project Leaders/PI's" section of the Terms of Use

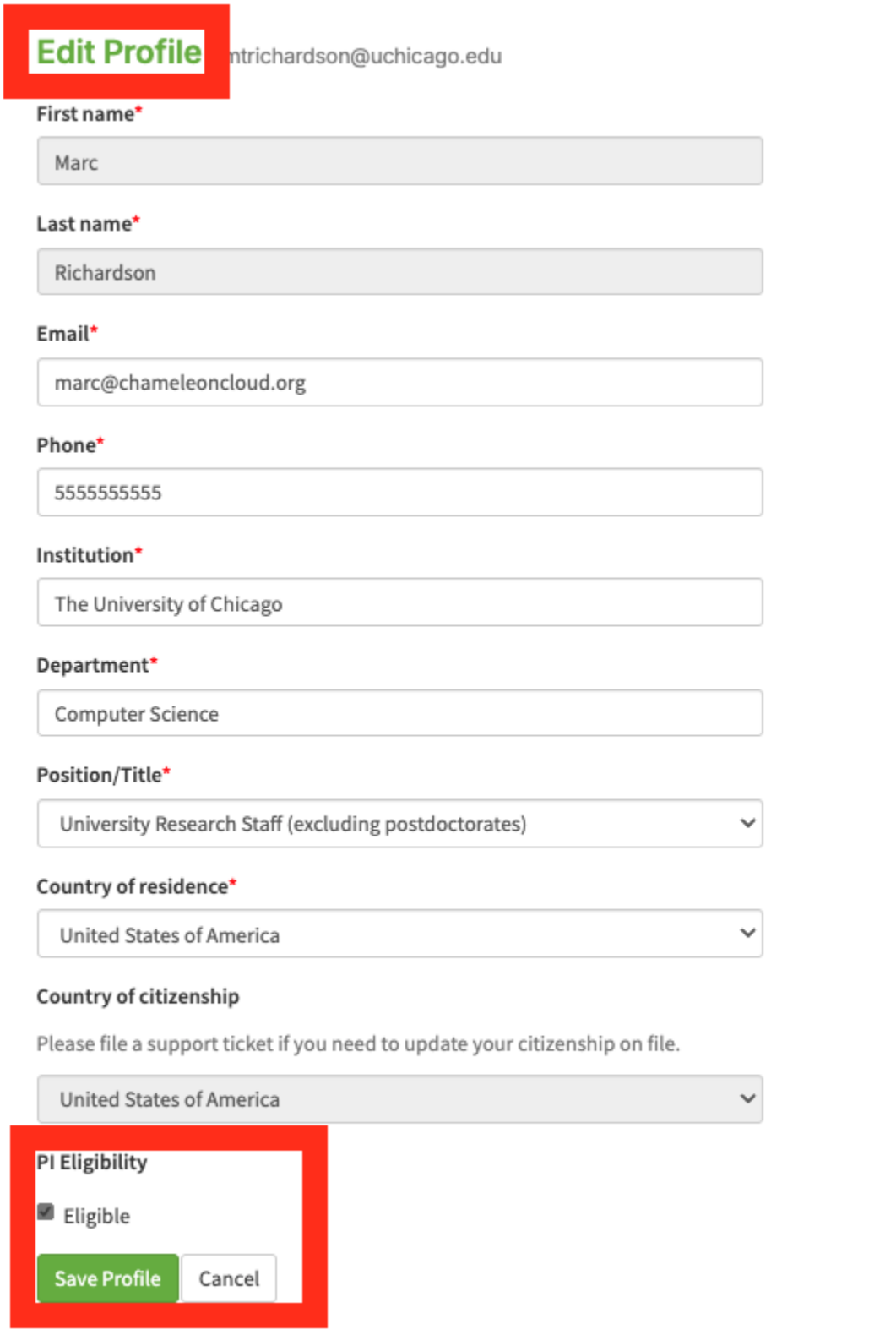


Fig. 3: Edit profile page

CHAPTER

NINE

GETTING HELP

Chameleon offers several support channels to assist users with their questions and issues. Depending on the nature of your inquiry, you can choose from the following options:

9.1 Community forum

> **Note**
>
> For immediate assistance or urgent issues, use the Help Desk rather than the community forum. The Help Desk ensures your request receives prompt attention from Chameleon staff.

Join the Chameleon Community Forums to connect with other users, ask questions, and share experiences. The forum provides a space for:

- User discussions and community support
- Non-urgent questions about using Chameleon
- Sharing best practices and experimental approaches
- Announcements of new features and capabilities

The forum is actively monitored by Chameleon staff and experienced community members who provide assistance on a best-effort basis. For immediate help or urgent issues, use the Help Desk instead.

9.2 Outages

The Outages page of the Dashboard contains a list of system outage announcements. You may subscribe to an RSS feed of these outages by clicking on the *RSS* icon.

9.3 Help desk

The Help Desk allows you to submit help request tickets and view the status of any open tickets.

To create a new help ticket, click the +Create a new ticket button and fill in the form. A system administrator will respond to your ticket within 3 business days.

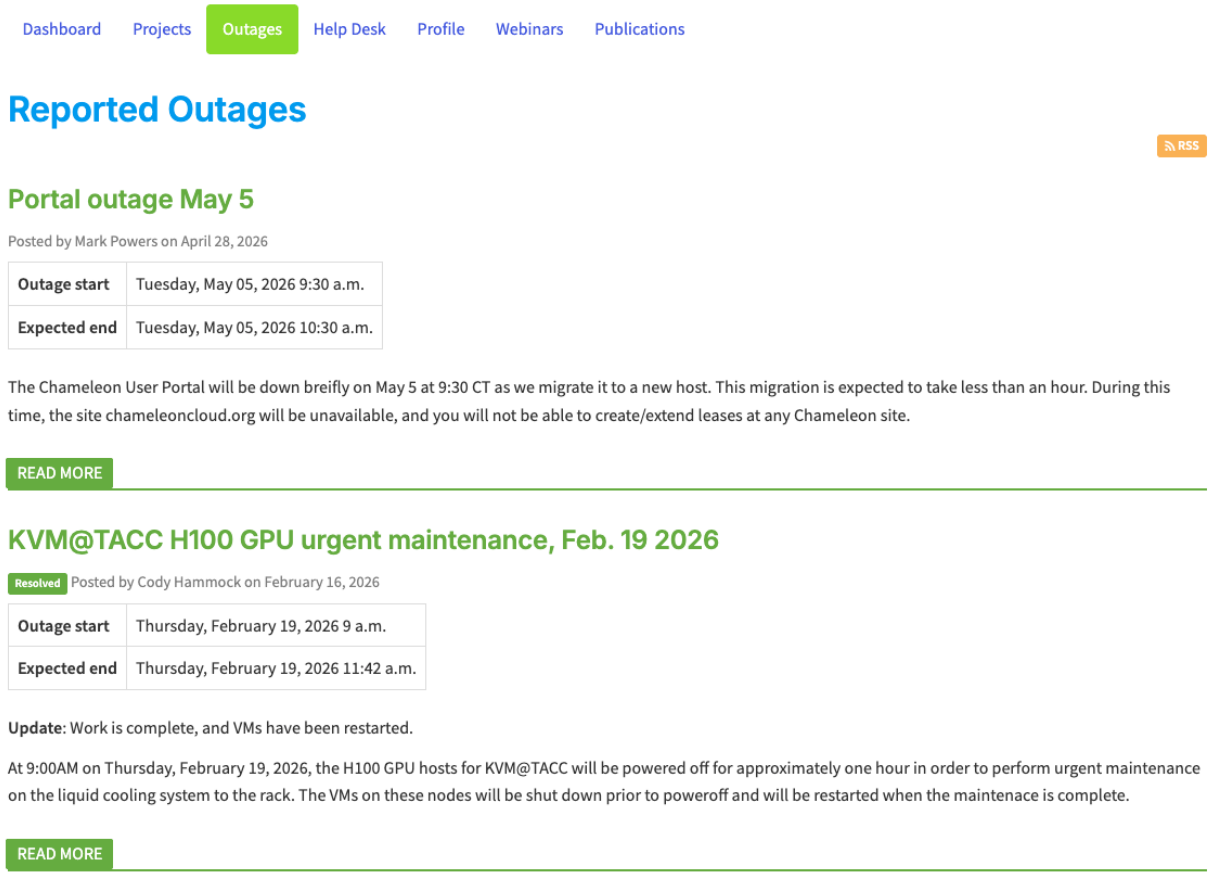


Fig. 1: The Outages announcement page

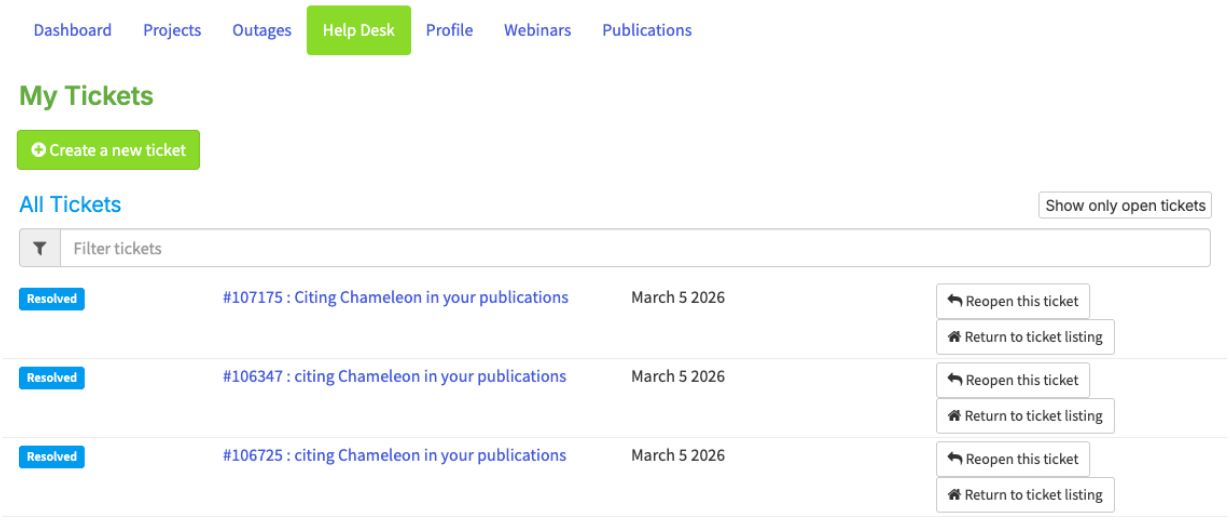


Fig. 2: The Help Desk page

9.4 Webinars

The Webinars page provides a list of upcoming webinars for Chameleon user training.

CHAPTER

TEN

GRAPHICAL USER INTERFACE (GUI)

The Graphical User Interface (GUI) provides a point-and-click experience for working with Chameleon resources. From the GUI, you may perform tasks such as manage and launch instances, and configure custom networking. Additionally, you may download an *OpenStack RC* file from the GUI if you wish to work with the *Command Line Interface*, instead. The Chameleon GUI is built on top of OpenStack Horizon (running OpenStack Antelope). Chameleon has multiple resource sites, each with its own URL (though it is possible to easily switch from one to other, see *Project and site menu*).

- CHI@TACC - Texas Advanced Computing Center: https://chi.tacc.chameleoncloud.org
- CHI@UC - University of Chicago: https://chi.uc.chameleoncloud.org
- CHI@NCAR - National Center for Atmospheric Research: https://chi.ncar.chameleoncloud.org
- CHI@Edge - Edge computing testbed: https://chi.edge.chameleoncloud.org (docs)

Chameleon also hosts KVM@TACC, a traditional OpenStack cloud where you may work with virtual machines. This site **does not** have access to bare metal resources. It is available at: https://kvm.tacc.chameleoncloud.org.

This section provides an overview of GUI interface navigation and basic functionality. For detailed instructions on using specific Chameleon features, see the dedicated documentation sections: *Bare Metal Instances*, *Networking*, *Reservations*, *Images*, *Object Store*, and *Complex Appliances*.

You may login to either site using your Chameleon portal username and password.

Attention

Each Chameleon testbed site—CHI@TACC, CHI@UC, CHI@NCAR, CHI@Edge, and KVM@TACC—is **independent**, so snapshots, keypairs, Swift containers, and other objects are unique to each site. For example, a keypair created at the CHI@TACC site is **not** available at the CHI@UC site. In addition, the bare metal resource types vary between sites.

10.1 GUI features

Upon logging in to the GUI at a Chameleon site, you will see your project's Overview page.

10.1.1 Project and site menu

To switch among the projects you belong to, use the project and site menu—the dropdown on the upper left of the screen next to the Chameleon logo. You can also use this menu to switch from one Chameleon site to another. This allows you to easily perform multi-site experiments.

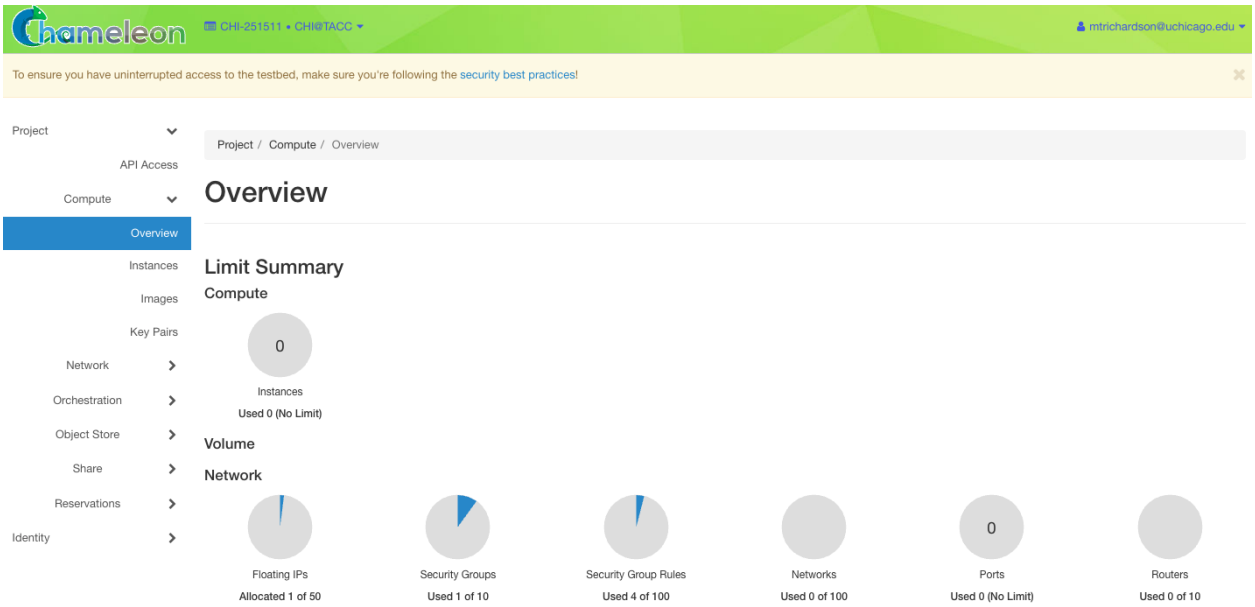


Fig. 1: The Chameleon GUI

10.1.2 User menu

To access user specific settings and download *OpenStack RC* files, use the user menu—the dropdown on the upper right of the screen where you will see your account name.

Settings

In the settings menu, you can change user specific settings such as the Timezone.

> **Note**
>
> Updating your timezone is **highly** recommended. When you make reservations for bare metal resources, your local time will be used. UTC is the default Timezone.

Documentation

The *Documentation* menu item will take you to this documentation site.

Help desk

The *Help Desk* menu item will take you to the Chameleon Help Desk, where you can submit support tickets and find answers to common questions.

OpenStack RC file

Clicking on this menu items will download a customized RC file for use with the OpenStack Command Line Interface. Source the RC file using `source` command to configure environment variables that allow you to easily log in using the *Command Line Interface*. For more information about *OpenStack RC* script, please see *The OpenStack RC script*.

Fig. 2: Switching between projects

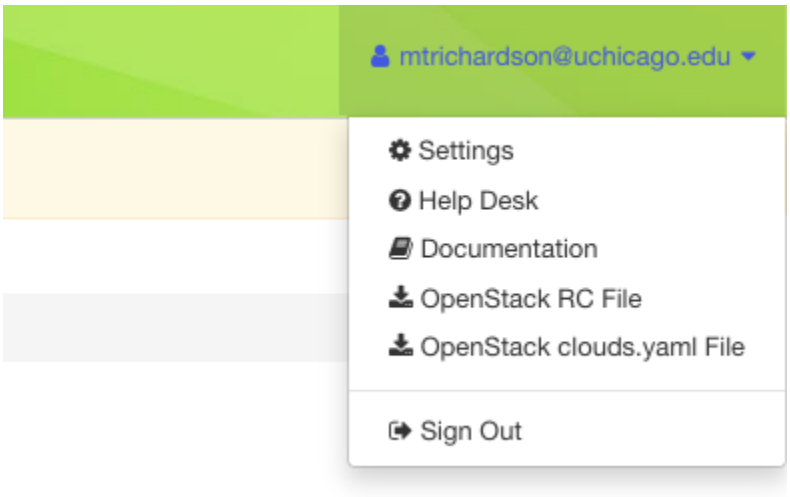


Fig. 3: The user dropdown menu

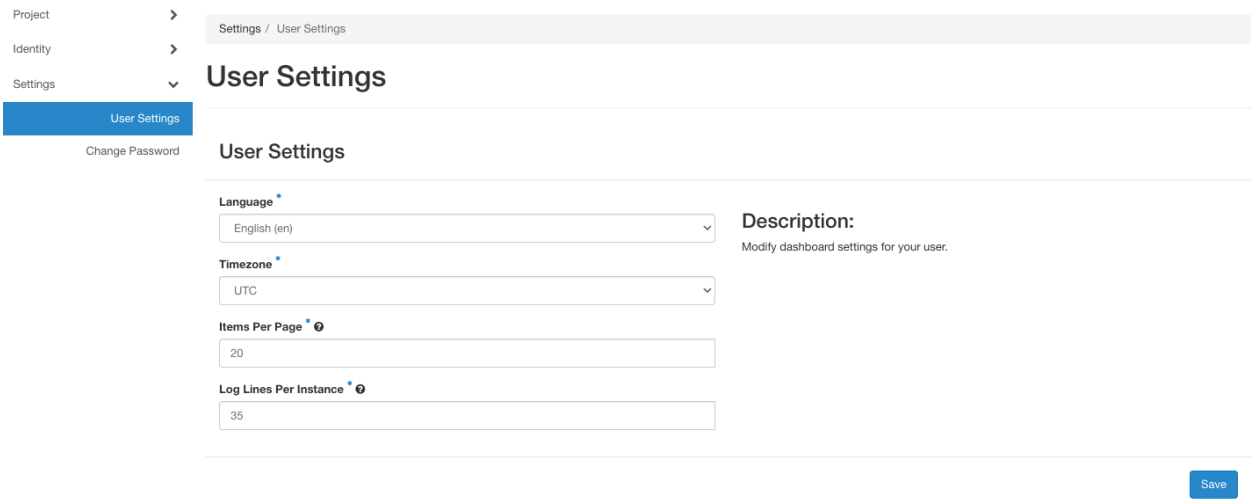


Fig. 4: User settings

OpenStack clouds.yaml file

Clicking on this menu item will download a `clouds.yaml` configuration file for use with the OpenStack Command Line Interface. This file can be used as an alternative to the RC file and allows you to manage multiple OpenStack environments in a single configuration file. See the OpenStack clouds.yaml documentation for details on how to use it.

Sign out

Use the *sign out* menu item to sign out from your current site.

> **Note**
>
> If you do not sign out manually, your session will expire in one hour.

10.2 API access

The API Access page lists all the available REST APIs that are used for configuring the *Command Line Interface (CLI)*. In addition, you may download *The OpenStack RC script* scripts via this page.

> **Note**
>
> Typically, the key generated from your computer will be at `~/.ssh/id_rsa.pub`. On Mac OS X, you can run in a terminal: `cat ~/.ssh/id_rsa.pub | pbcopy`. It copies the content of the public key to your copy/paste buffer. Then you can simply paste in the "Public Key" box.

10.3 GUI navigation

The navigation sidebar on the left allows you to access different sections of the interface. The main navigation elements are described below.

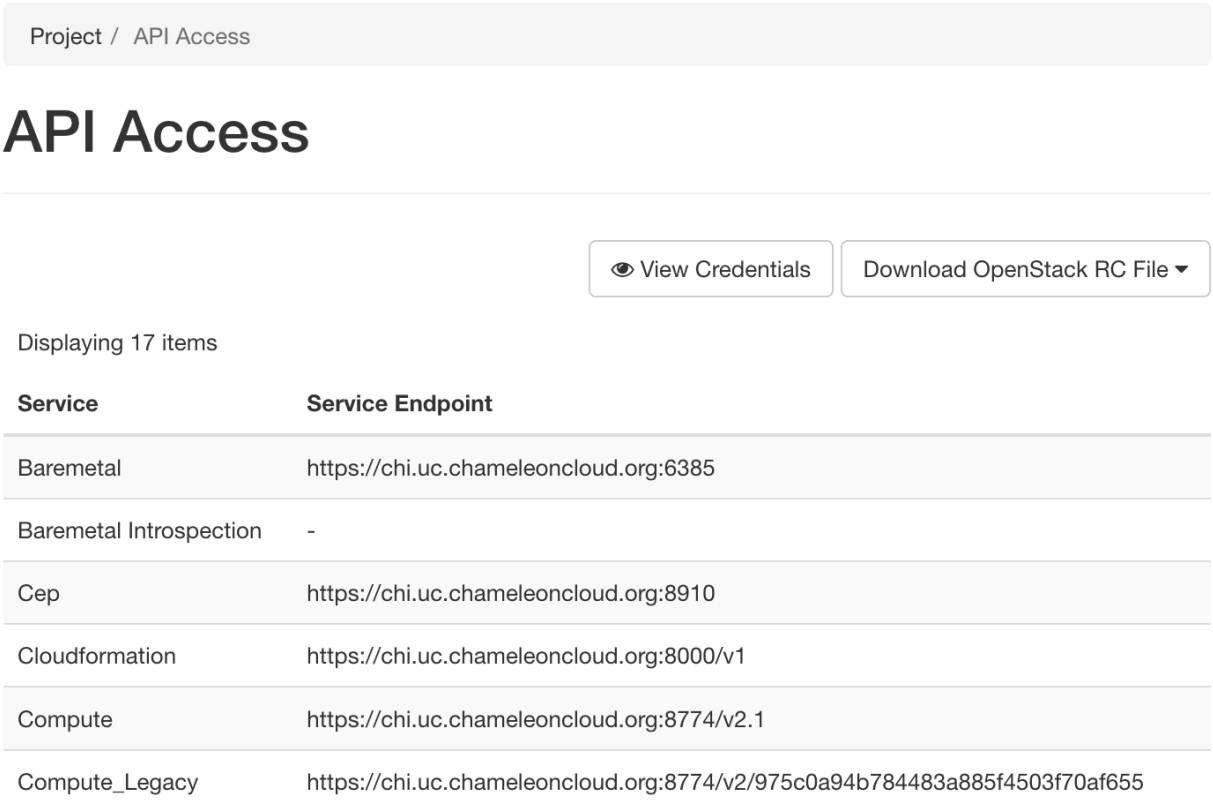
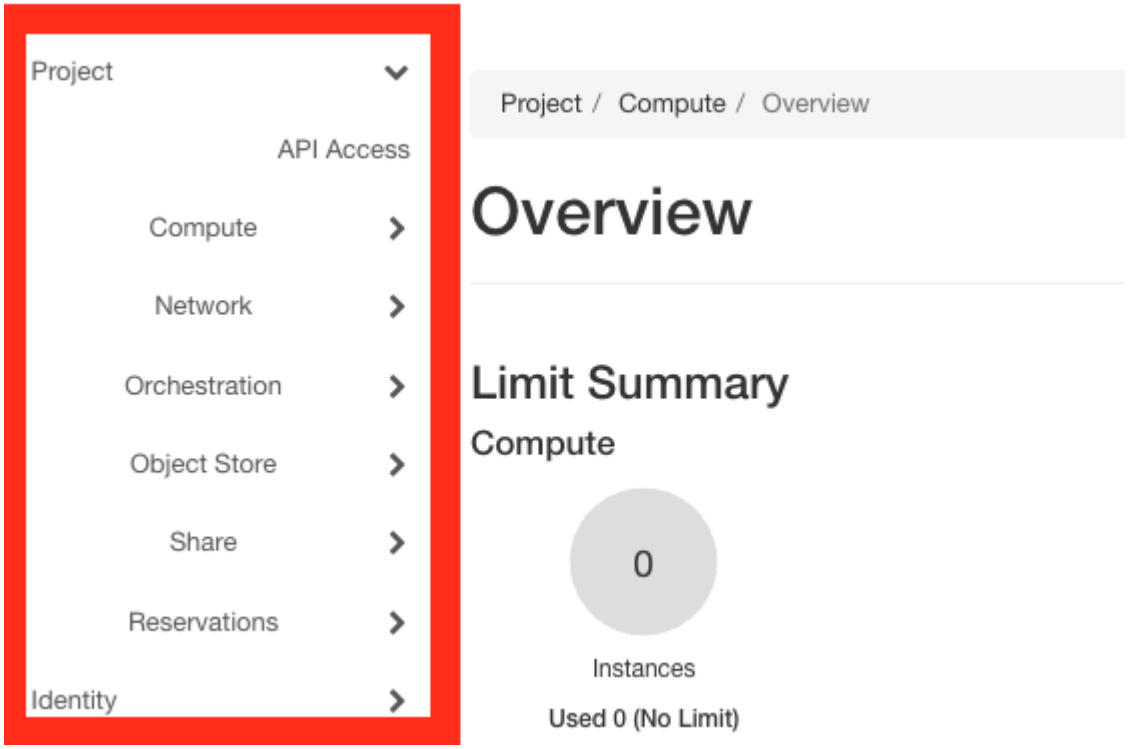


Fig. 5: The API Access page

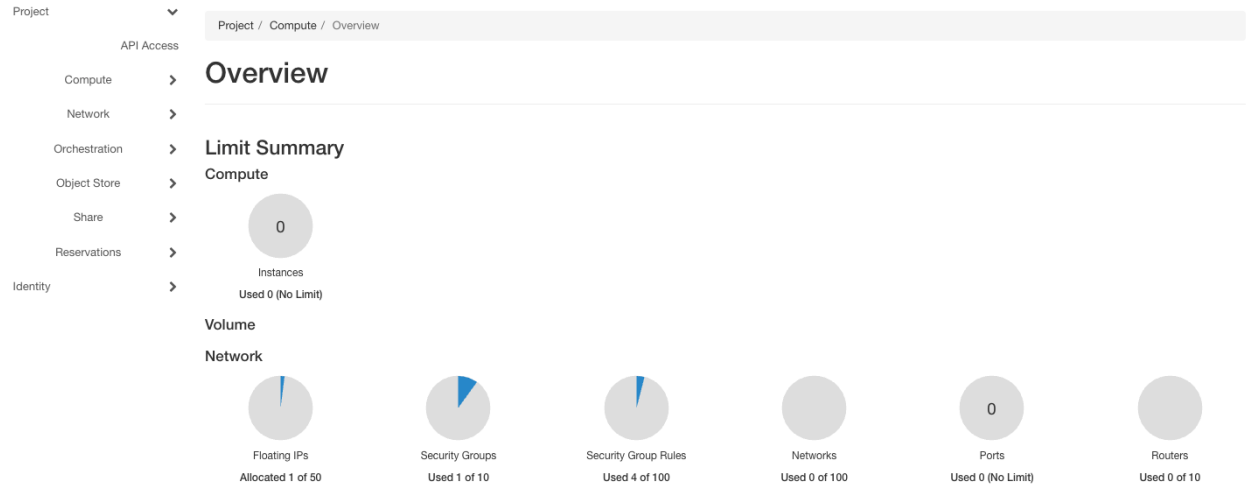


10.3.1 Compute

The *Compute* section provides interfaces for managing instances, images, and SSH key pairs. It is backed by OpenStack Nova.

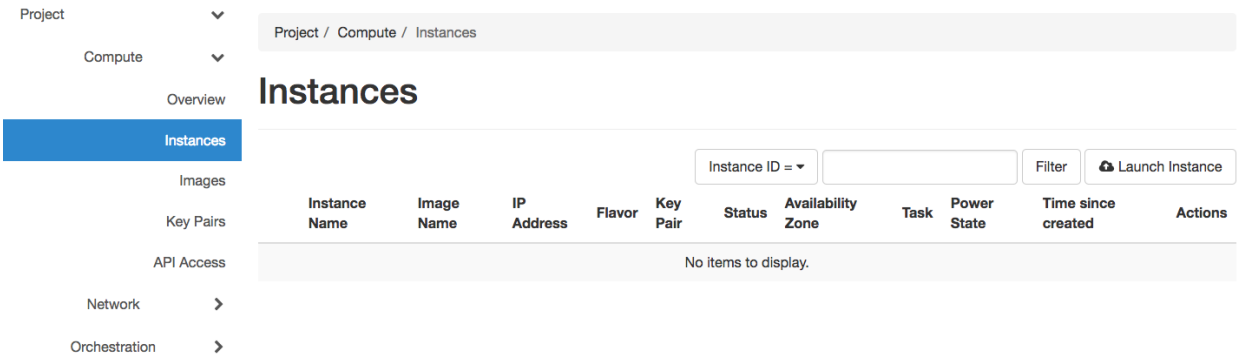
Overview

The Overview page provides a graphical summary of your project's current resource usage.



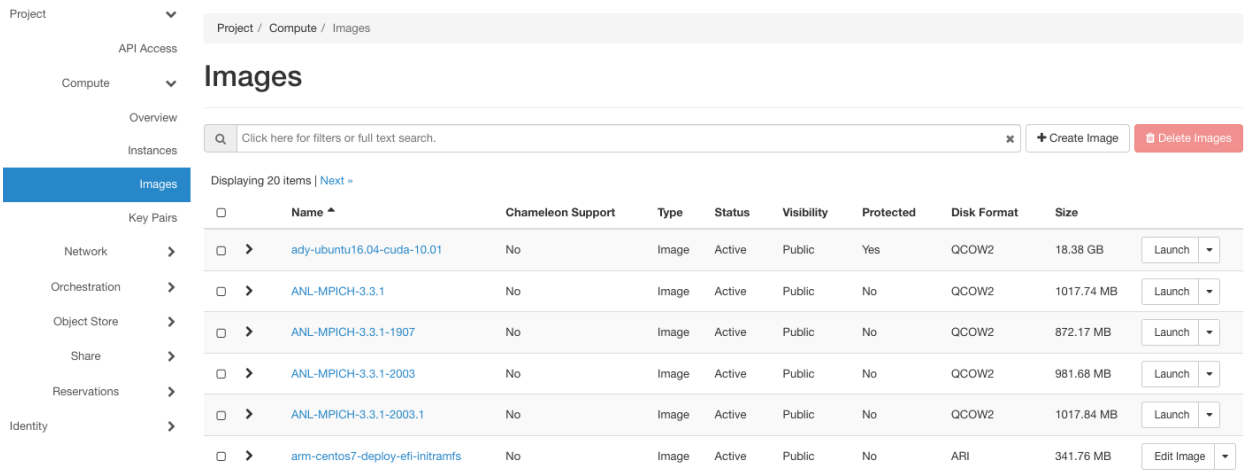
Instances

The *Instances* page displays your running instances with options to launch, terminate, monitor, or reboot them. For detailed instructions on launching and managing instances, see *Bare Metal Instances*.



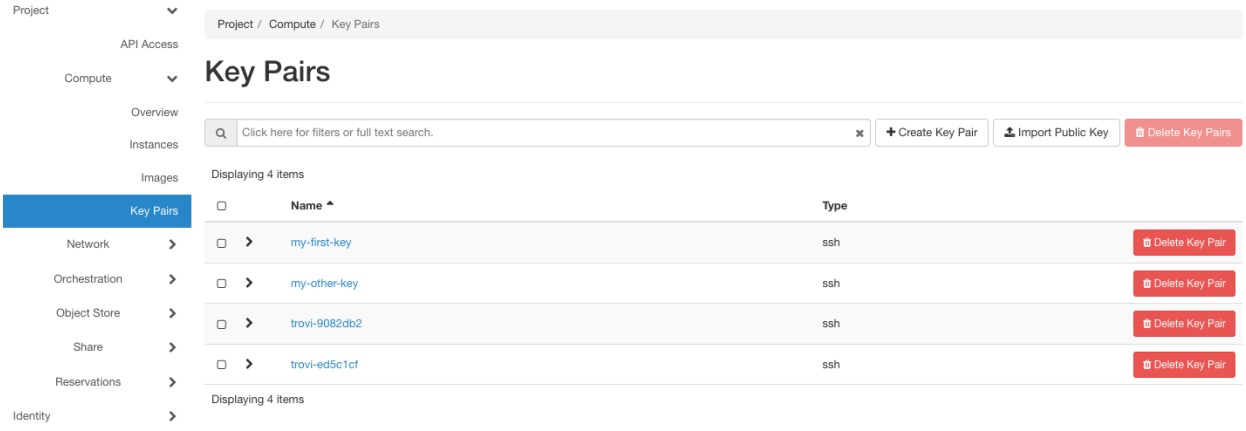
Images

The *Images* page allows you to view available images and launch instances from them. Images are managed by OpenStack Glance. You can only edit images you own. For comprehensive image management including uploading and sharing, see *Images*.



Key pairs

The *Key Pairs* page allows you to create, import and manage SSH key pairs for instance access.



For detailed instructions on creating and importing key pairs, see the guide here.

10.3.2 Network

The *Network* section provides interfaces for managing virtual network resources. It is backed by OpenStack Neutron. For comprehensive networking instructions, see *Networking*.

Network topology

The *Network Topology* page displays your current virtual network topology in topology or graph formats.

Networks, routers, and floating IPs

The *Networks*, *Routers*, and *Floating IPs* pages allow you to create and manage these network resources for your project.

Attention

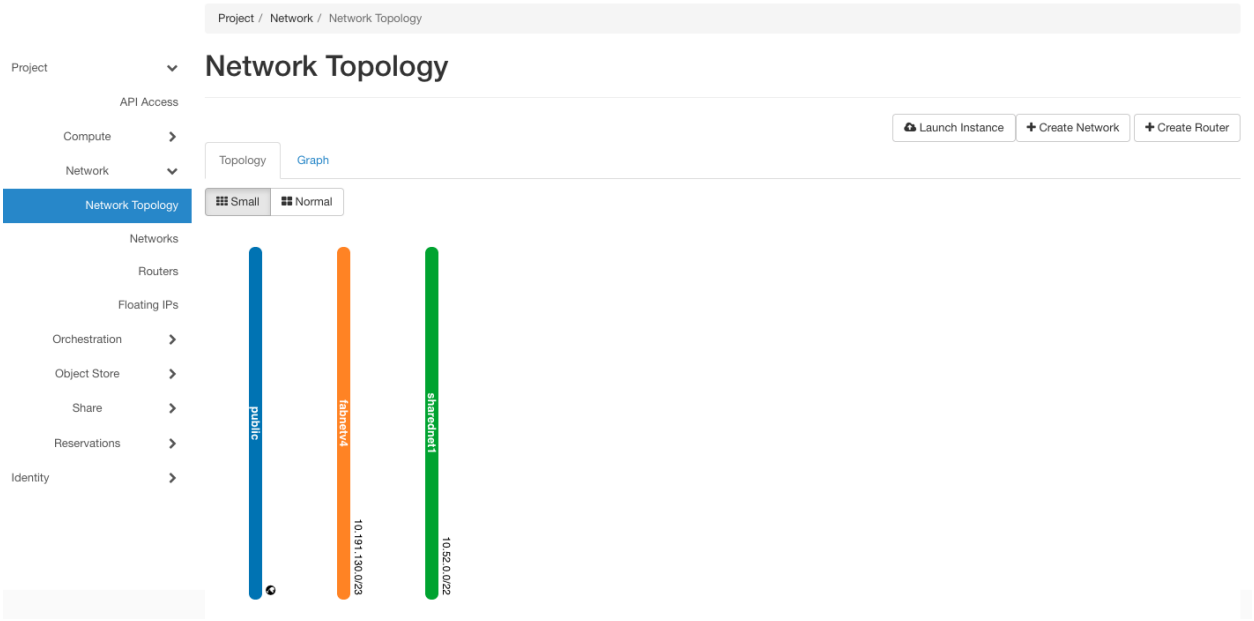
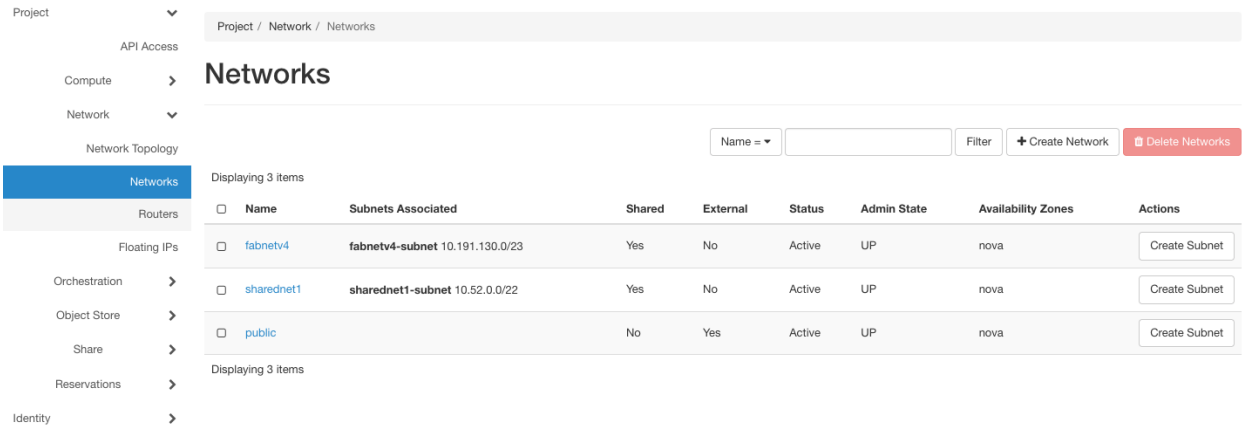


Fig. 6: The Network Topology page



Chameleon bare metal sites (CHI@TACC, CHI@UC, CHI@NCAR) **do not** support security groups - all ports are open to the public.

10.3.3 Orchestration

The *Orchestration* section provides interfaces for working with complex appliances and Heat templates. It is backed by OpenStack Heat. For comprehensive instructions, see *Complex Appliances*.

Stacks

A deployed complex appliance is referred to as a "stack" – just as a deployed single appliance is typically referred to as an "instance". The Stacks page allows you to launch, rebuild, or terminate stacks.

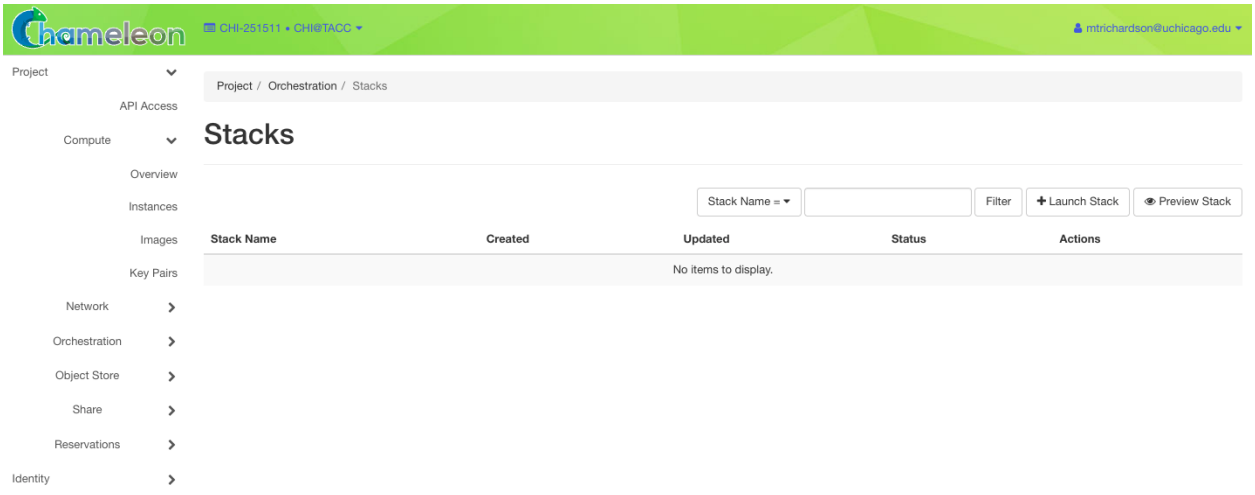


Fig. 7: The Stacks page

Resource types

The *Resource Types* page lists all Heat resource types available for use in your templates, along with their properties and attributes.

Template versions

The *Template Versions* page lists the supported Heat Orchestration Template (HOT) versions and the features available in each.

Template generator

The *Template Generator* provides a graphical interface for building Heat templates without writing YAML by hand. For more on writing templates, see *Heat orchestration templates*.

10.3.4 Object store

The *Containers* section provides access to Chameleon's object/blob storage, backed by OpenStack Swift. For detailed object store instructions, see *Object Store*.

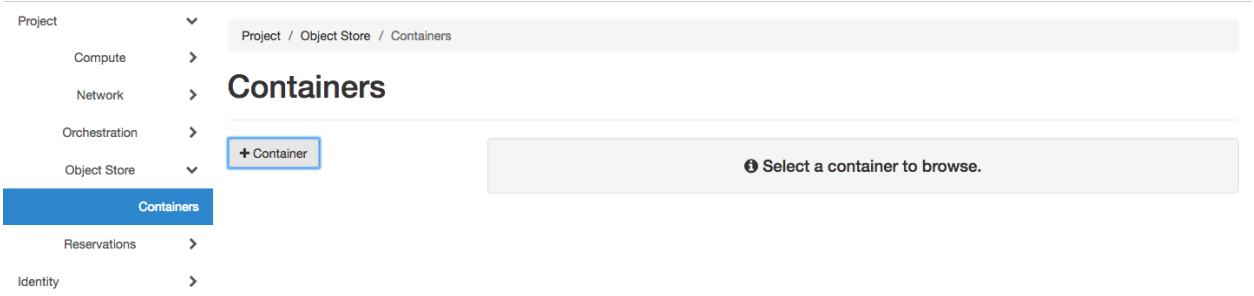
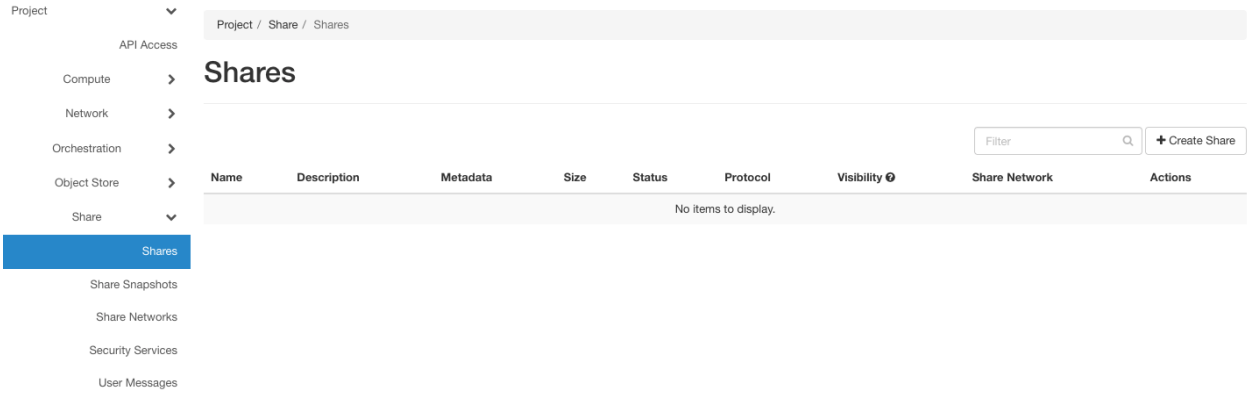


Fig. 8: The Containers page

10.3.5 Share



The *Share* section provides interfaces for managing shared file systems using OpenStack Manila. For detailed instructions, see *Shares*.

Shares

The *Shares* page allows you to create and manage shared file systems that can be mounted by multiple instances simultaneously. For background on share concepts and step-by-step procedures, see *Shares concepts* and *View share*.

Share snapshots

The *Share Snapshots* page allows you to create point-in-time snapshots of your shares for backup or cloning purposes.

Share networks

The *Share Networks* page allows you to configure the network settings that shares use to communicate with instances.

Share groups

The *Share Groups* page allows you to group shares together so that consistent snapshots can be taken across multiple shares at once.

Security services

The *Security Services* page allows you to configure authentication services (such as Active Directory or LDAP) that can be associated with share networks.

10.3.6 Reservations

The *Reservations* section allows you to manage your resource leases. It is backed by OpenStack Blazar, Chameleon's bare metal reservation service. For comprehensive instructions, see *Reservations*.

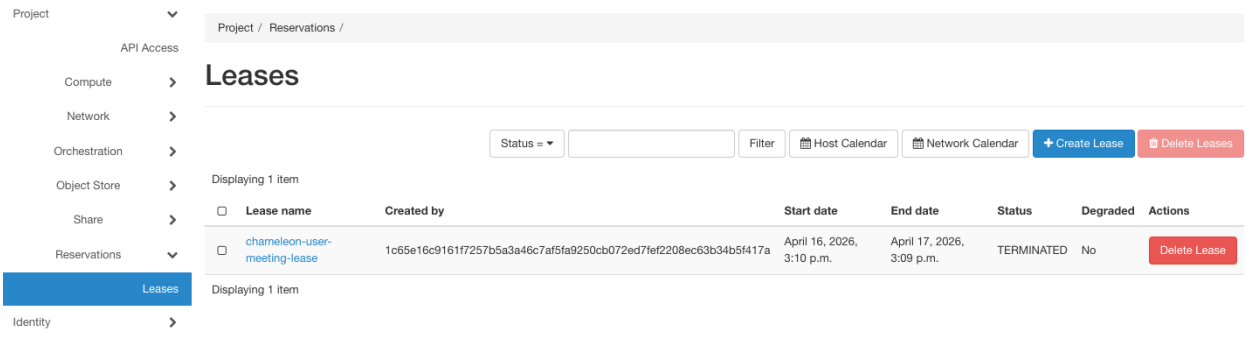


Fig. 9: The Leases page

10.3.7 Identity

The *Identity* section provides interfaces for managing your account's projects, users, and credentials. It is backed by OpenStack Keystone.

Projects

The *Projects* page shows projects you belong to and allows you to set your default project. For more on managing project membership and allocations, see *Project Management*.

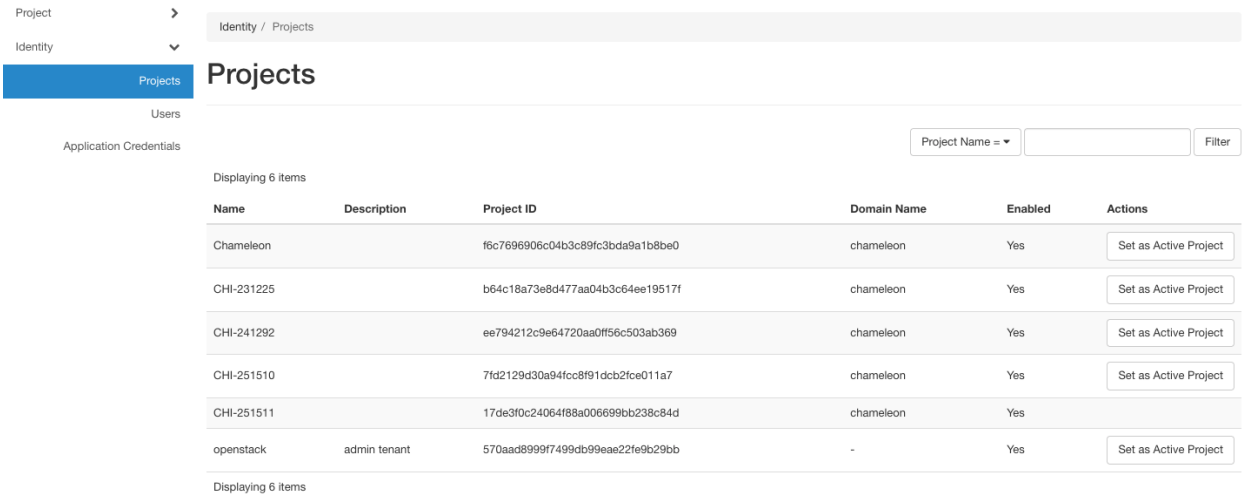


Fig. 10: The Projects page

Users

The *Users* page displays your account information. You can use this page to view your user ID, which is sometimes required when managing project membership or configuring access controls.

Application credentials

The *Application Credentials* page allows you to create scoped credentials for use by applications or scripts without exposing your primary account password. Application credentials are tied to a specific project and can be given a limited set of roles. For instructions on using application credentials with the CLI, see *Creating an application credential*.

CHAPTER

ELEVEN

COMMAND LINE INTERFACE (CLI)

The Command Line Interface (CLI) provides a way to interact with Chameleon resources using shell and scripting tools. Chameleon uses the OpenStack Client to provide CLI functionality. This documentation section provides an overview on how to install the client and configure your shell environment to access Chameleon features.

Which authentication method should I use? We recommend *ccauth* for most users: it authenticates via a browser-based device flow and can generate credentials for multiple projects and sites at once, so you don't need to manually download and manage separate credential files. If you're working with a single project and site and want a tool with nothing to install, *cc-login* is a lighter-weight option preinstalled on CC-* images. The *password and application credential methods* still work but require manually creating and downloading credential files for each project — only reach for them if neither ccauth nor cc-login fits your workflow.

Looking to script or orchestrate experiments in Python instead of shell? python-chi is Chameleon's Python library and offers a programmatic alternative to the CLI — see our *Jupyter and python-chi guide* for an introduction.

Attention

Some of the Chameleon features are **only** accessible via the CLI, such as power monitoring tools and the advanced networking features.

Chameleon Cloud is primarily designed to support Unix-like environments. Therefore, it is highly recommended using CLI in a Unix-like system. For Windows 10 users, you may want to enable Windows Subsystem for Linux to get better experience with the Chameleon CLI.

11.1 Installing the CLI

11.1.1 Prerequisites

1. **Python** - Check if you have Python installed.
2. **pip** - If you're using Python 3.4 (or greater), then pip comes installed with Python by default.

11.1.2 OpenStack client installation

1. Install the CLI by typing `pip install python-openstackclient chameleon-blazarclient` in the terminal.
2. Verify that it has installed correctly by typing `openstack`. You will enter the OpenStack Client in interactive mode and your prompt should change to `(openstack)`.
3. Exit the client by typing `exit`.

11.2 ccauth command

`ccauth` is a command-line tool that authenticates you with Chameleon via your browser and generates OpenStack credentials (`clouds.yaml` or `openrc`) for use with the OpenStack CLI.

The `ccauth` package must be installed in the same Python environment as the OpenStack clients using it. On first run, you'll be prompted to visit a URL to authenticate via your browser. Subsequent runs reuse the cached refresh token silently, unless it has expired and needs to be refreshed.

11.2.1 Installation

`ccauth` is preinstalled and available in the PATH on all Chameleon supported images, so no installation is required when using Chameleon images.

If you are installing `ccauth` on your local system or from the source repository, install it using pip from the GitHub repository:

```
pip install git+https://github.com/ChameleonCloud/ccauth.git
```

11.2.2 Quick start

Generate a `clouds.yaml` file for the current site and project:

```
# On first run you'll be prompted to visit a URL to authenticate.
ccauth clouds-yaml --output ~/.config/openstack/clouds.yaml

# Use the cloud entry
export OS_CLOUD=openstack
openstack server list
```

Subsequent runs reuse the cached refresh token silently. The `ccauth login` command is available as a convenience to pre-authenticate or verify credentials, but it is not required — the `clouds-yaml` and `discover-projects` commands will trigger the device flow automatically if no cached token is present.

11.2.3 Usage

```
ccauth [OPTIONS] COMMAND
```

11.2.4 Commands

ccauth login

Optional. Runs the OIDC device flow and caches a refresh token. Useful for pre-authenticating or verifying credentials before running other commands. Discovers the current site from the OpenStack metadata service when on a Chameleon instance.

```
ccauth login
ccauth login --auth-url https://chi.uc.chameleoncloud.org:5000/v3
ccauth --debug login
```

ccauth logout

Clears the cached refresh token.

```
ccauth logout
```

ccauth clouds-yaml

Writes a `clouds.yaml` file. By default, generates a single entry for the current site and project (discovered from the OpenStack metadata service). Use `--all-sites` to cover all Chameleon sites.

```
# Current site + current project (default)
ccauth clouds-yaml --output ~/.config/openstack/clouds.yaml

# All sites + current project, one entry per site
ccauth clouds-yaml --output ~/.config/openstack/clouds.yaml --all-sites

# Current site + all projects, one entry per project
ccauth clouds-yaml --output ~/.config/openstack/clouds.yaml --all-projects

# All sites + all projects
ccauth clouds-yaml --output ~/.config/openstack/clouds.yaml --all-sites --all-projects

# Overwrite existing entries
ccauth clouds-yaml --output ~/.config/openstack/clouds.yaml --force
```

The `--all-projects` flag generates one entry per (site, project) pair, named `<site>_<project>`. The `--no-vendordata` flag skips the metadata service lookup; it requires `--auth-url` or `--all-sites`.

ccauth openrc

Writes an `openrc` file for a single site (use `clouds-yaml` for multi-site).

```
ccauth openrc --output ~/openrc
ccauth openrc --auth-url https://chi.uc.chameleoncloud.org:5000/v3 --output ~/openrc
ccauth openrc --output ~/openrc --force
```

ccauth discover-projects

Lists all projects you have access to across all sites and prints ready-to-run `ccauth clouds-yaml` commands for each one. Triggers the device flow automatically if no cached token is present.

```
ccauth discover-projects
ccauth discover-projects --output ~/my-clouds.yaml   # use a custom path
```

Example output:

```
Found 2 project(s). To add a project to clouds.yaml, run:

  # Project 1
  ccauth clouds-yaml --all-sites --project-id abc123 --output ~/.config/openstack/clouds.
↪yaml

  # Project 2
```

(continues on next page)

(continued from previous page)

```
 ccauth clouds-yaml --all-sites --project-id def456 --output ~/.config/openstack/clouds.
↪yaml
```

11.2.5 Examples

Complete workflow: authenticating and running OpenStack commands

To authenticate and run commands like `openstack image list`, use the `clouds.yaml` approach:

```
ccauth clouds-yaml --output ~/.config/openstack/clouds.yaml
export OS_CLOUD=openstack
openstack image list
```

> **Note**
>
> `ccauth` is available as a library for developers who wish to use it in their own projects, or for those interested in contributing to its development. For more information, see the ccauth GitHub repository.

Other examples

List all your projects and available authentication options:

```
ccauth discover-projects
```

Pre-authenticate or verify credentials:

```
ccauth login
```

Generate credentials for all Chameleon sites:

```
ccauth clouds-yaml --output ~/.config/openstack/clouds.yaml --all-sites
```

11.2.6 Site discovery

Without `--auth-url`, the current site comes from the OpenStack metadata service at `169.254.169.254` (vendordata). This works automatically when running on a Chameleon instance.

With `--all-sites`, `ccauth` fetches all available sites from the Chameleon reference API (https://api.chameleoncloud.org/sites) and merges in the current site from vendordata when available (e.g., to pick up the correct cloud name for KVM).

Use `--no-vendordata` to skip the metadata service entirely; this requires `--auth-url` or `--all-sites`.

Override the discovery endpoints with `--sites-api-url` and `--metadata-url`.

11.3 cc-login command

If you use a CC-* image, a `cc-login` command will be available in the `cc` user's *PATH*. The command performs device authentication, caches an application credential for reuse, and can generate OpenStack `openrc` files or entries for `clouds.yaml` so the OpenStack CLI and libraries can use the cached credential.

The device authentication flow directs you to a URL in your browser where you log in with your Chameleon credentials and approve the device request. Once approved, `cc-login` creates an application credential on your behalf and caches it for reuse, so subsequent OpenStack operations do not require authentication.

Warning

Authentication information is no longer provided through instance vendor-data — see *Changes to automatic credential setup on instances* for details. Use `cc-login` to obtain fresh application credentials instead.

Note

`cc-login` can be used to generate credentials for a single project and Chameleon site and does not require a custom plugin for OpenStack authentication. To generate credentials for multiple projects or sites, via device flow authentication with OIDC tokens that automatically refresh, see *ccauth*.

11.3.1 Usage

```
cc-login [OPTIONS]
```

11.3.2 Options

The command accepts various options to configure authentication, credential caching, and output format. Some of the most commonly used ones are:

- `--app-cred-name`, `--app-cred-expires-hours`: Name and lifetime (in hours) for the application credential (default: 24 hours).
- `--ttl-seconds`: Local cache validity period in seconds before cc-login will refresh the credential (default: 24 hours).
- `--force-refresh`: Bypass the local cache and perform fresh device authentication + app-credential creation.
- `--output-openrc`, `--force-openrc`: Write an `openrc`-style file with the app-credential (useful for sourcing in shell sessions). Use `--force-openrc` to overwrite an existing file.
- `--output-clouds-yaml`, `--cloud-name`, `--force-clouds-yaml`: Add or update a cloud entry in `clouds.yaml` (default cloud name: `chameleon`). Use `--force-clouds-yaml` to overwrite an existing entry.

11.3.3 Examples

Complete workflow: authenticating and running OpenStack commands

To authenticate and run commands like `openstack image list`, you have two options to use credentials obtained via `cc-login`:

Option 1: Source an openrc file

Generate an `openrc` file and source it in your shell session:

```
cc-login --output-openrc ~/openrc_chameleon
source ~/openrc_chameleon
openstack image list
```

Option 2: Use clouds.yaml

Alternatively, generate or update an entry in `clouds.yaml` and use the `OS_CLOUD` environment variable:

```
cc-login --output-clouds-yaml ~/.config/openstack/clouds.yaml \
  --cloud-name chameleon
export OS_CLOUD=chameleon
openstack image list
```

Other examples

Force a fresh device authentication (ignoring the local cache):

```
cc-login --force-refresh
```

11.4 CLI authentication

When using the CLI, you have to provide some credentials so the system trusts that the operations are really being executed by your user account. This page covers two such ways of doing this: a CLI password and application credentials. Both require manually creating and downloading credential files.

Note

For most users, *ccauth* or *cc-login* are easier than the methods on this page — they authenticate via your browser and generate credential files for you, with no manual password or application credential setup required. See *which authentication method should I use?* for guidance on choosing between them. Use a CLI password or application credential only if neither of those tools fits your workflow.

11.4.1 Setting a CLI password

You can set a CLI password via the Chameleon Authentication Portal. The password you associate with your account can not be used to log in to the GUI or Jupyter interfaces and can only be used to authenticate a command-line client.

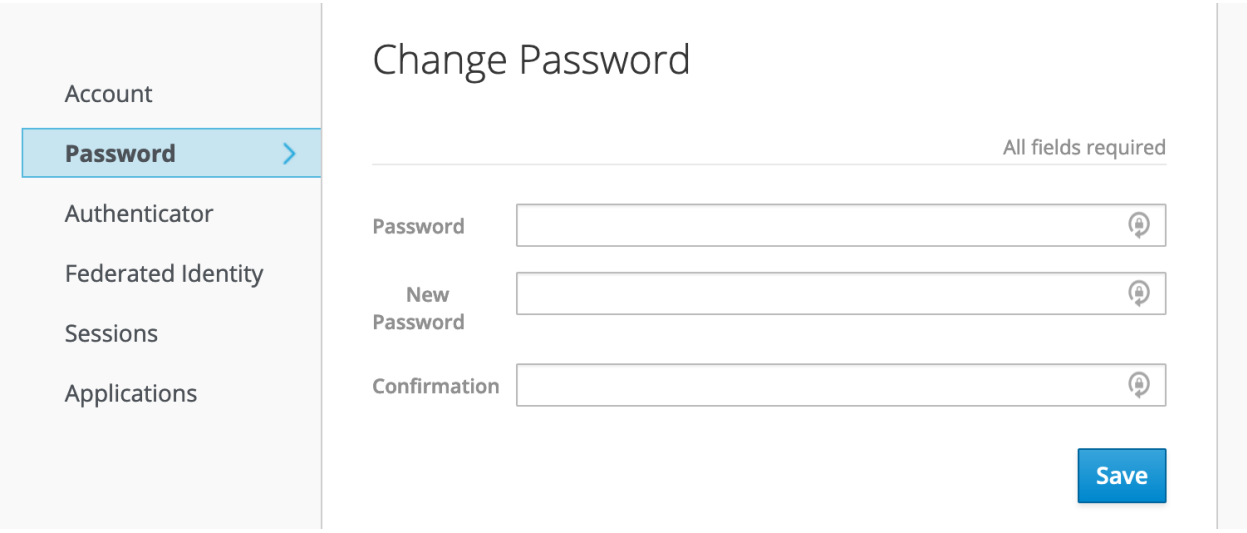


Fig. 1: Setting a password in the Chameleon Authentication Portal

The benefit of this method is that this password will work on any Chameleon site.

Note

You should set a strong password for your CLI password, and it should not be a password you use elsewhere. Otherwise, your account risks being compromised by an attacker who has possibly obtained your password from another breached service. We **highly** recommend using a password manager e.g., BitWarden, LastPass, or 1Password to assist.

11.4.2 Creating an application credential

You can also generate *application credentials*, which act as dedicated one-off passwords that are authorized with a scoped set of your user account's permissions, within a single project. If you work on multiple projects simultaneously, you will need to generate one application credential for each project.

To create an application credential, navigate to the "Identity" dashboard in the *Graphical User Interface (GUI)*, and go to the "Application Credentials" panel. Create a new application credential and name it something meaningful (such as "CLI access for project CH-XXX"). **You will also need to check the "unrestricted" checkbox in order to use the CLI to make leases in Blazar**. If you do not need to make reservations via the CLI, you can leave the box unchecked, as it is the safer option.

Note

If a credential scoped only to the `member` role gets a `403 Forbidden` error on commands such as `openstack image list` or `openstack server create`, try creating a new application credential that also includes the `reader` role alongside `member`. This is a known issue with role scoping for application credentials; including `reader` resolves it for most operations while a permanent fix is in progress.

Once the system generates the credential, you will be given the option to download an *RC file* that configures the CLI to use the application credential for authentication. You will only see the secret credentials once, so make sure to save the RC file or the secret somewhere, as if it's lost, you will have to delete the credential and create a new one.

11.5 The OpenStack RC script

You must use the *OpenStack RC Scripts* to configure the environment variables for accessing Chameleon features. You can downloaded the script from the Chameleon GUI at the *API access*.

Note

On an instance running a CC-* image, you can use the *cc-login* or *ccauth* command to generate a new openrc file or clouds.yaml entries. This is useful for creating fresh credentials without downloading them from the Chameleon GUI and copying them to your instance — and is now the primary way to get credentials on an instance, since they are *no longer provisioned automatically at boot*.

1. Log in to the GUI at on a CHI site.

   Important

   Download the RC file from the site you would like to interact with. The RC files are different for each site.

2. Select the project you wish to access via *Project and site menu*.
3. Download *OpenStack RC Script* using *User menu* by clicking on *Openstack RC File*.

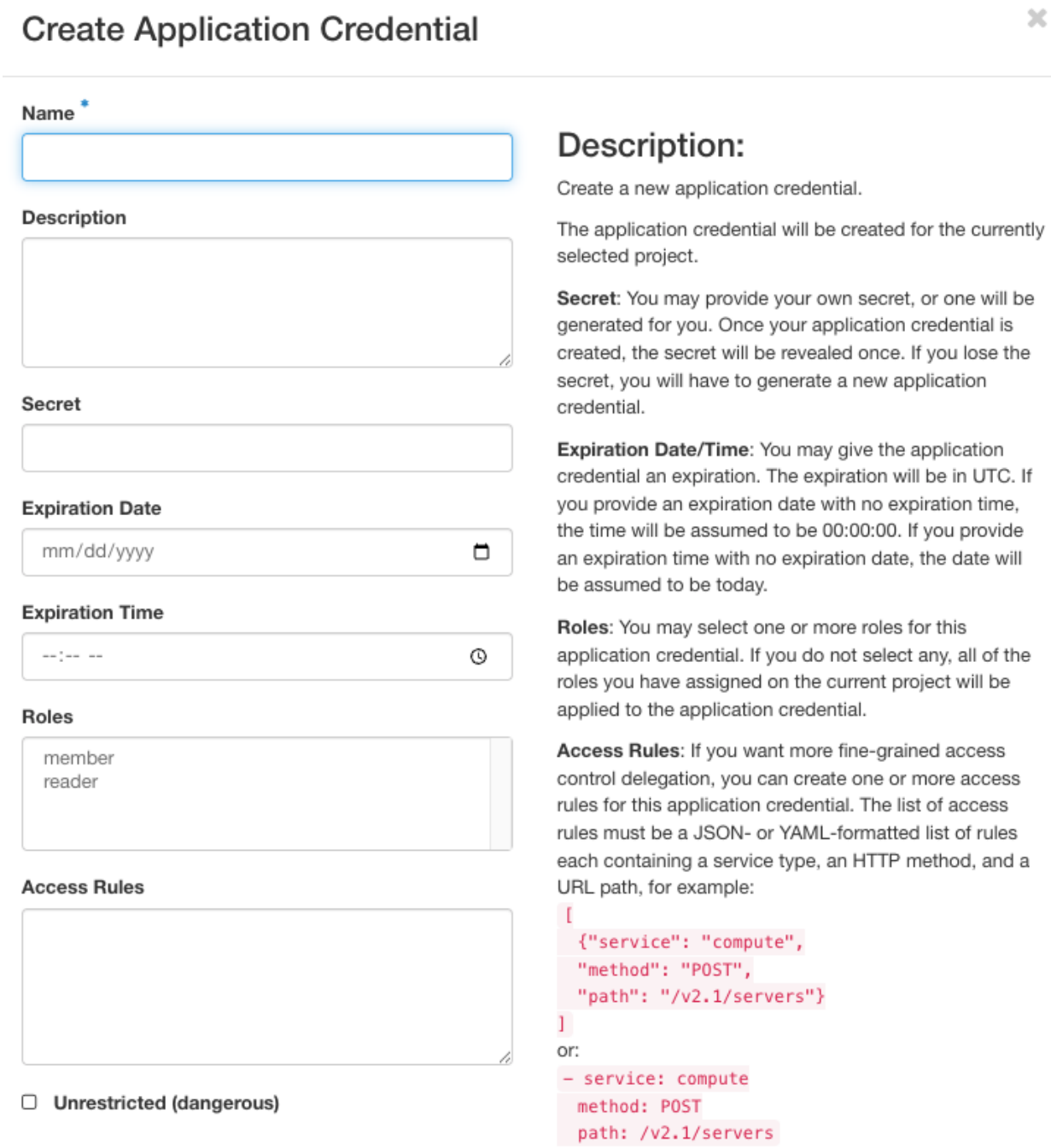

Create Application Credential
Name *
Description
Secret
Expiration Date
mm/dd/yyyy
Expiration Time
--:-- --
Roles
member
reader
Access Rules
Unrestricted (dangerous)
Description:
Create a new application credential.
The application credential will be created for the currently selected project.
Secret: You may provide your own secret, or one will be generated for you. Once your application credential is created, the secret will be revealed once. If you lose the secret, you will have to generate a new application credential.
Expiration Date/Time: You may give the application credential an expiration. The expiration will be in UTC. If you provide an expiration date with no expiration time, the time will be assumed to be 00:00:00. If you provide an expiration time with no expiration date, the date will be assumed to be today.
Roles: You may select one or more roles for this application credential. If you do not select any, all of the roles you have assigned on the current project will be applied to the application credential.
Access Rules: If you want more fine-grained access control delegation, you can create one or more access rules for this application credential. The list of access rules must be a JSON- or YAML-formatted list of rules each containing a service type, an HTTP method, and a URL path, for example:
[
{"service": "compute",
"method": "POST",
"path": "/v2.1/servers"}
]
or:
- service: compute
method: POST
path: /v2.1/servers

Fig. 2: The Project Dropdown

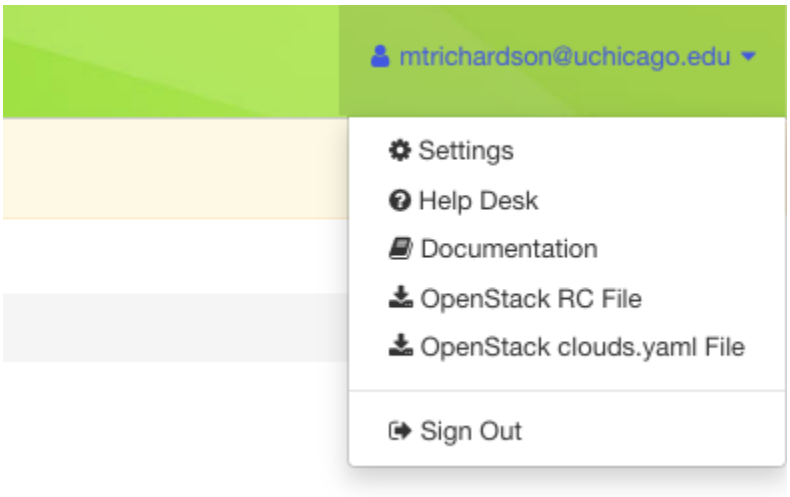


Fig. 3: The OpenStack RC File link in the User Dropdown

> **Tip**
>
> As an alternative to the RC file, you can also download a `clouds.yaml` file from the same menu. This lets you manage credentials for multiple OpenStack environments in a single configuration file. See the OpenStack clouds.yaml documentation for details.

4. Run the following command in the terminal:

```
source <path/to/openstack_rc_file>
```

> **Note**
>
> The command **will not** work for Windows users. Skip this step and the next step if you are using Windows system.

5. Enter your password when prompted.

> **Note**
>
> This prompt is for your *CLI password*, set via the Chameleon Authentication Portal — **not** the institutional, Google, or ORCiD credentials you use for federated login. If you don't have a CLI password yet, see *CLI authentication* to set one, or use *ccauth* instead to avoid needing one at all. If you sourced an RC file generated from an *application credential*, no password prompt will appear, since the credential secret is already embedded in the file.

6. For macOS/Linux users, your current terminal session has been configured to access your project. Now type `openstack` in your terminal session.

   For Windows users, you have to provide the environment variables in the *OpenStack RC* script as `openstack` command parameters. Run the following command in your Windows prompt:

```
openstack --os-auth-url <OS_AUTH_URL> \
--os-project-id <OS_PROJECT_ID> \
--os-project-name <OS_PROJECT_NAME> \
--os-user-domain-name <OS_USER_DOMAIN_NAME> \
--os-username <OS_USERNAME> \
--os-password <OS_PASSWORD> \
--os-region-name <OS_REGION_NAME> \
--os-interface <OS_INTERFACE> \
--os-identity-api-version <OS_IDENTITY_API_VERSION>
```

   Replace values of the parameters by reading from the *OpenStack RC* script.

   Another way to configure the OpenStack client for Windows users is to add/edit environment variables manually via *System Properties* window. Then, click on *Environment Variables...* button and manually add/edit the environment variables in *OpenStack RC Script* to *Environment Variable* window.

> **Note**
>
> For macOS/Linux users, every time when open a new terminal, you have to run the `source` command to access the OpenStack client.

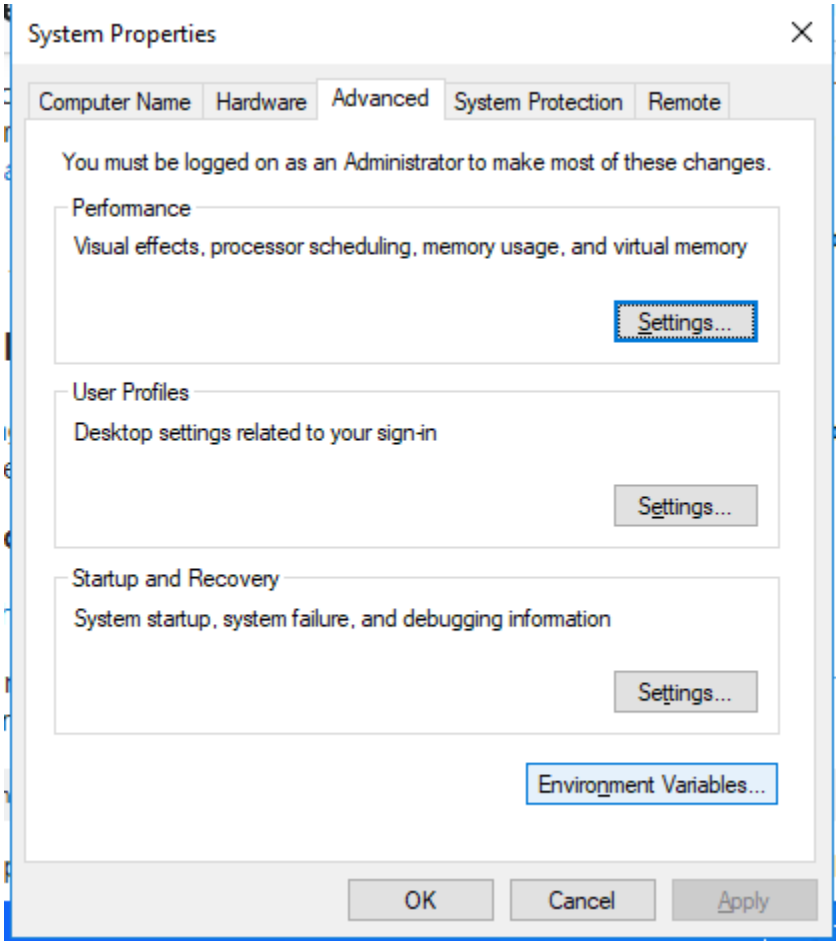


Fig. 4: System Properties Window of Windows System

Error

If you get authentication error, check if you input your password correctly.

7. Type `project list` at the `(openstack)` prompt. You should see a list of the projects you belong to.

Error

If you get permission error at this step, check that:

- the terminal session has been configured correctly with the environment variables
- the *OpenStack RC* script you `source` is **v3**
- the OpenStack client version is the latest. To check the OpenStack client version, use `openstack --version` command. Some older versions may cause errors.

Error

If you get the `Missing value` error when using a command, it is likely that your terminal session has not been configured correctly and completely with the environment variables. The error may be fixed by re-running the `source` command over the OpenStack RC Script or using the command line switches.

11.6 Changes to automatic credential setup on instances

Chameleon's vendordata service previously (June 2026 and earlier) provided authentication information to instances at boot, which Chameleon-supported (CC-*) images used to automatically populate `~/openrc` and mount the object store. This authentication information has been removed from vendordata.

Important

Instances no longer come with OpenStack credentials or an object store mount configured automatically. You must now authenticate with *ccauth* yourself after the instance boots.

11.6.1 Authenticating with the CLI

`~/openrc` is no longer pre-populated. Generate it yourself with `ccauth`, which authenticates via OIDC device flow instead of vendordata:

```
ccauth openrc --output ~/openrc
source ~/openrc
openstack image list
```

See *ccauth* for the full set of commands, including `clouds-yaml` for multi-site or multi-project credentials.

Note

The `cc-generate-openrc` command, which previously read credentials from vendordata to populate `~/openrc`, has been removed. Use `ccauth openrc` instead.

11.6.2 Mounting the object store

The object store is no longer auto-mounted at boot. To set it up, authenticate with `ccauth` and then run `setup-cc-mount-object-store` once:

```
ccauth openrc --output ~/openrc
setup-cc-mount-object-store
```

11.6.3 Changes to `cc-snapshot`

`cc-snapshot` no longer looks up credentials from vendordata. It now always uses OpenStack credentials from the environment (`clouds.yaml`/ `OS_CLOUD` or `OS_AUTH_URL`/`OS_TOKEN`), and must be run with `sudo -E` so root inherits them:

```
ccauth openrc --output ~/openrc
source ~/openrc
sudo -E cc-snapshot <image_name>
```

See the *cc-snapshot utility* page for full usage.

11.6.4 Older images

These changes affect vendordata itself, so they apply regardless of which image an instance is running. Older images that predate this change are affected too, but cannot fall back to the tools they previously relied on:

- `cc-generate-openrc` will no longer work, since vendordata no longer carries credentials. Obtain an openrc file via *ccauth* or by downloading it from the Horizon GUI instead.
- `cc-snapshot` will require OpenStack credentials to be set in the environment, as described above.
- Any object store mount configured to run at boot will fail, since the credentials it depended on are no longer available. You'll need to get the credentials and mount the object store manually as described above.

11.7 Working with resources via the CLI

> **Tip**
>
> Reading *CLI authentication* and *which authentication method should I use?* is highly recommended before continuing on the following sections.

The Chameleon CLI is built on the OpenStack Client, along with a few service-specific clients (Blazar, Manila, Heat, Swift) for features beyond core compute and networking. Rather than duplicate command references here, this page covers the modes you can run the CLI in, and then maps out where each task is actually documented, alongside the equivalent GUI instructions.

11.7.1 Command modes

You can use the CLI in either Interactive Mode or Shell Mode. In either mode, the OpenStack client has to be configured by using the *OpenStack RC Script* or by providing the command line switches. For more information about the usage of the OpenStack client, run `openstack --help`.

Interactive mode

The Interactive Mode allows you to use the `openstack` commands through an interactive prompt. To start the Interactive Mode, type `openstack` in the configured terminal. Once entering the Interactive Mode, you will see a `(openstack)` prompt. Type the command you would like to run at the prompt. To find out the commands, type `help`.

Shell mode

Each CLI command can be used in your terminal exactly the same way that it appears in the Interactive Mode, simply by preceding the command with `openstack`. For example, the command `image list` in the Interactive Mode is equivalent to the command `openstack image list` in the Shell Mode.

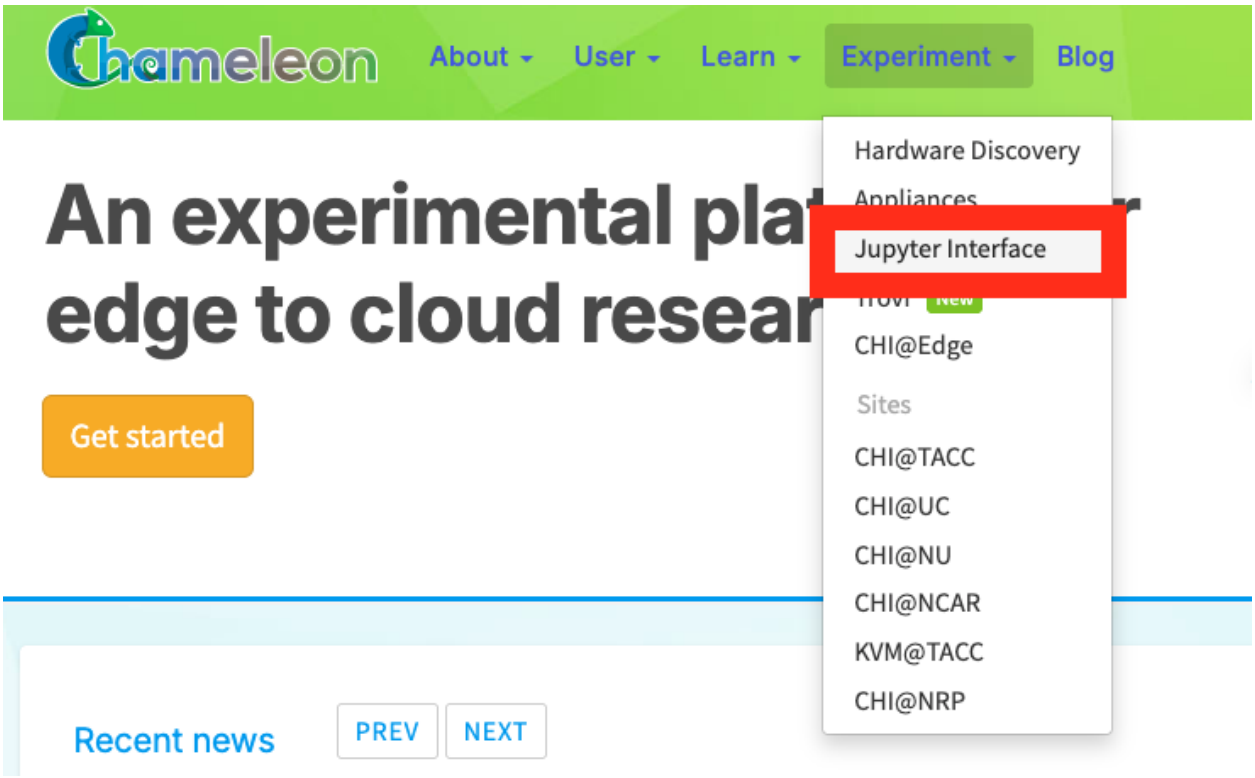
11.7.2 Common tasks

- **Discover hardware and resources**: resource discovery is available via the GUI and the *REST API* — there is no dedicated `openstack` CLI command for browsing hardware, see *Resource Discovery* for details.
- *Reserve resources (leases)*: create and manage Blazar leases for bare metal nodes, network segments (VLANs), and floating IPs.
- *Launch and manage bare metal instances*: create, list, and delete bare metal servers.
- *Work with KVM instances*: KVM-specific CLI tasks, such as converting images to raw format for better launch performance.
- *Manage images and snapshots*: upload, list, and snapshot images.
- *Manage the object store*: upload and retrieve objects via Swift.
- *Manage shares*: create and manage shared file systems via Manila.
- *Manage complex appliances*: orchestrate multi-resource appliances via Heat.

CHAPTER

TWELVE

JUPYTER INTERFACE



Jupyter Notebooks are an excellent tool for prototyping, exploring, and ultimately documenting the entire experimental process. They combine the benefits of explanatory text, executable code, and rich visualization/interaction.

Chameleon users can get a Jupyter Notebook server automatically provisioned for them by logging in to the JupyterHub server managed by Chameleon. Upon login, you will be redirected to your Jupyter Notebook server. If there is not yet a Notebook server allocated for your user, one will be created behind the scenes. This can take a few moments.

Warning

The shared Jupyter environment places resource limits on your Jupyter server, notably limiting it to 1 CPU core and 1GB of memory. If you are doing computationally or memory-intensive work in a Notebook, it may be beneficial to look in to *provisioning your own dedicated JupyterHub*.

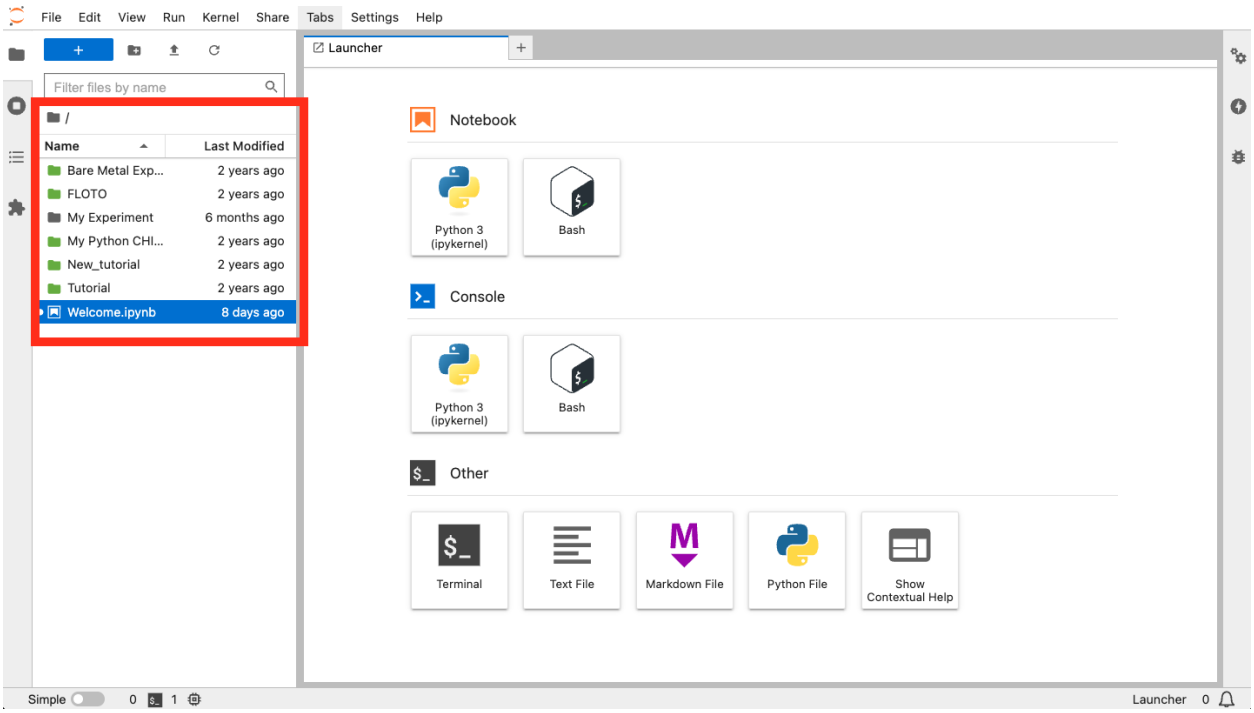
12.1 JupyterLab interface overview

When you are logged in, you will land in the JupyterLab application environment. For up-to-date documentation about the JupyterLab interface, see the official JupyterLab documentation.

You will see a file browser on the left-hand side - this is your working directory. It's yours, so feel free to create and delete files as you see fit. Your working directory is initially populated with a few examples to help you get started, such as an example Notebook. **Files that you save here will be persisted even if your server is torn down; the next time you log in the data will be restored**. You should consider the rest of your server environment ephemeral, as updates to the Jupyter interface can cause your server to be re-built.

Hint

Jupyter Notebooks do not deal well with large files, and you should avoid trying to edit large files in the interface as it can cause instability, slowness, or even crashes. If you need to deal with large files it is best to process them on a *dedicated processing node*, such as a baremetal node provisioned as part of your experiment.



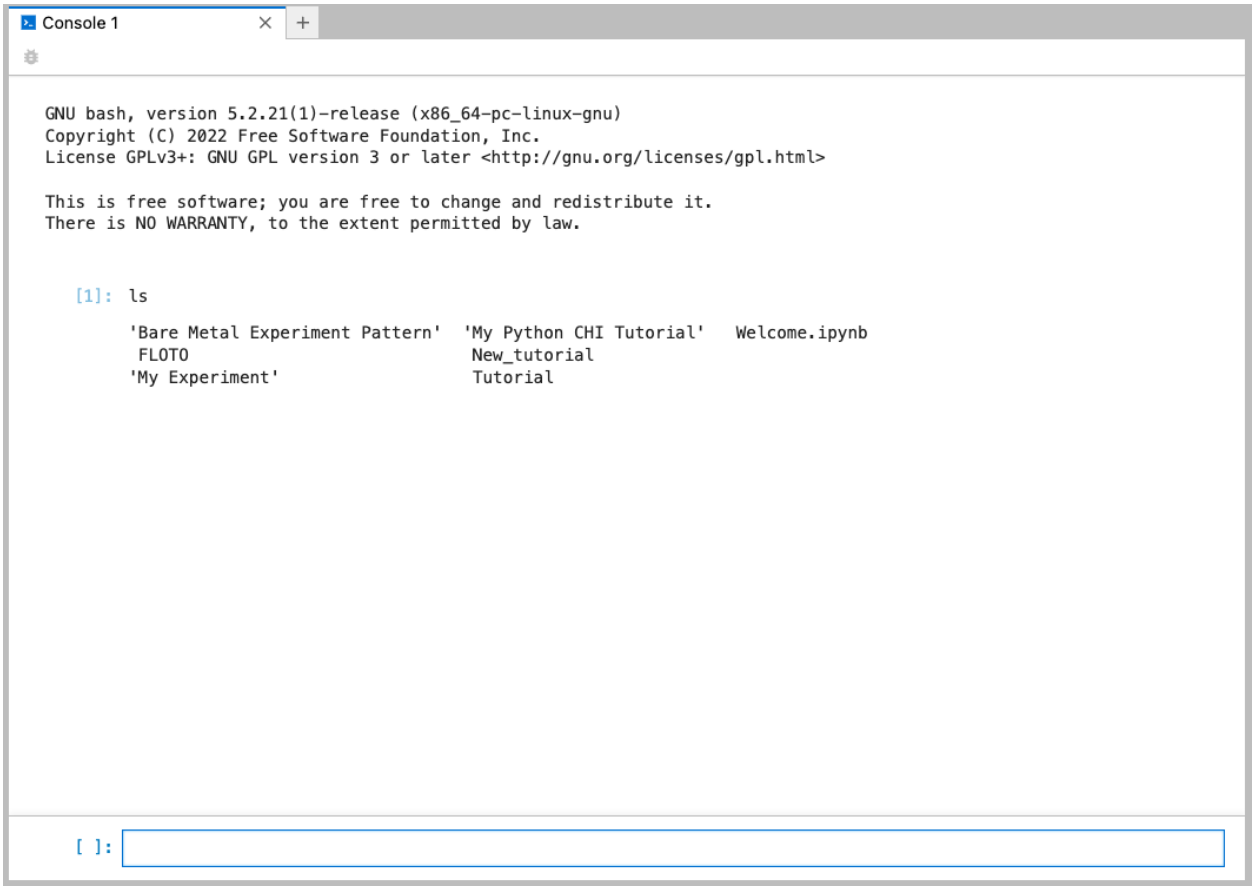
12.2 Working with notebooks

A great place to start is the Bare Metal Experiment Pattern on Trovi, which provides a complete, reusable notebook template for reserving, configuring, and running experiments on bare metal nodes. Beyond that, Trovi hosts a wide range of community notebook artifacts that demonstrate Chameleon features and make it easy to get hands-on with different resource types and workflows.

All Notebook servers come with OpenStack python clients installed as well as the `python-chi` library for programmatic interaction with Chameleon resources. Other python modules you may want to use in your Notebook can be installed via the *Console interface*.

12.3 Console interface

You can open a web terminal console by going to File > New > Terminal. This works just like a remote shell, and you will also have *sudo* access so you can install additional software to support your needs.



Hint

All Chameleon Notebook servers are built from a common base image. This means if your server is torn down (which can happen during an upgrade of the Jupyter server), you may have to re-do any changes to the underlying system you made since the server was created. For this reason it is a good idea to put this setup code in a script in your working directory. Your working directory is backed up and will persist across Jupyter server restarts.

12.4 Advanced topics

12.4.1 Dedicated Jupyter servers

The *default Jupyter environment* available to all Chameleon users is a bit limited: you are working within a shared environment and as such there are some practical limitations around the amount of CPU cores and memory you can utilize. More intensive analytical workflows may function better from within a dedicated Jupyter server for use by you and/or other members of your project.

Trovi provides a JupyterHub appliance you can instantiate on Chameleon. For a detailed walkthrough, see our blog post Running experiments inside a Jupyter Notebook.

12.4.2 Collaboration strategies

It is often desirable to share your in-progress Notebooks with peers or supervisors for feedback, perhaps before publishing in *Packaging & Sharing Experiments*. This can be accomplished in many different ways, each suiting different use-cases. We have identified a few current tools that offer the best range of functionality.

Trovi sharing portal

The easiest way to share and collaborate on a Notebook is to publish it to *Trovi*.

Pros

- Already integrated into Jupyter; no need to sign up or log in to anything else.
- No need to download and copy Notebooks and other data around.
- Supports sharing with other Chameleon users and projects.

Cons

- Limited support for real-time collaboration; last edit wins.
- No support (yet!) for sharing one-time or expiring links with collaborators outside of Chameleon.

Google Colaboratory

Google provides a free Jupyter Notebook execution environment that can run your Notebook files in a private VM on Google's cloud infrastructure. As it is a Google product, a Google account is required to use it. Notebooks can be edited by users concurrently, similar to functionality present in Google Docs. Notebooks are stored in Google Drive and as such can be easily shared using the existing Drive sharing mechanisms. Finally, and notably, hardware-accelerated computation via GPUs and TPUs is available for free exploration. For more details see the FAQ.

Pros

- Supports rich real-time collaboration on Notebook files.
- Notebooks easily sharable via Google Drive to others with Google accounts.
- Can manage access to private Notebooks via ACLs.
- Free to use.

Cons

- Not intended for long-running tasks. Your experiment may be terminated prematurely if it is deemed an invalid use of resources.
- Chameleon libraries not pre-installed. You can however install the Python API client to your Notebook via the special `!pip install python-chi` syntax. See the Importing Libraries example notebook for examples on how to install new libraries.

- Requires Google account.

GitHub + Nbviewer

A common pattern that works for many use-cases is using GitHub as the backing store for your Notebooks. This is nice because you get version history for free due to Git VCS being used behind the scenes. GitHub Notebooks are easily sharable (you just send a link) and there is decent support in GitHub for viewing the current state of the Notebook and its rendered outputs. To allow others to actually run your Notebook, you can either import the Notebook files back in to your Chameleon JupyterLab instance, or use Binder, which allows spinning up a Jupyter instance for a given GitHub link.

Pros

- Supports version history via Git VCS.
- Supports easily sharing rendered Notebooks (read-only) via GitHub links.
- Can import the Notebook into a personal Jupyter server (such as the one provided by Chameleon) or via a hosted tool like Binder.
- Changes can be proposed using Pull Request workflows you may already be familiar with.

Cons

- Running the Notebook requires getting it into a Jupyter server somehow.
- Requires GitHub account if you want to keep your Notebooks private.
- Services like Binder don't create Jupyter servers with Chameleon tools (like the `python-chi` Python API) built in by default.

CHAPTER

THIRTEEN

OVERVIEW

The following sections contain in-depth knowledge for utilizing Chameleon's advanced features.

Note

Most workflows described in this guide can also be orchestrated programmatically with python-chi, Chameleon's Python library — see our *Jupyter and python-chi guide* for an introduction and the module reference for full API details.

- *Resource Discovery*: Discover Chameleon bare metal resources by node type and view node information.
- *Reservations*: Reserve Chameleon resources for use in your Project.
- *Bare Metal Instances*: Launch and manage Instances on Chameleon bare metal resources. This is a core feature of Chameleon.
- *Images*: Create images of Instances.
- *Power Monitoring*: Monitor power consumption and energy usage of your experiments.
- *Complex Appliances*: Work with Complex Appliances, which automate the process of deploying multiple Instances with reconfigurable networking.
- *Object Store*: Store user data such as files as Objects in portable Containers.
- *Shares*: Provide file storage to an instance with the OpenStack Shared File System service.
- *Networking*: Create Isolated virtual networks within Chameleon.
- *FPGAs*: Configure and work with FPGA nodes.
- *Packaging & Sharing Experiments*: Package and share reproducible experiments and artifacts using the Trovi Sharing Portal, including granting temporary Daypass access to collaborators or reviewers.
- *KVM*: Use non-bare metal virtual machine resources in Chameleon's OpenStack implementation.
- CHI@Edge: Use container-based edge computing resources, such as Raspberry Pi devices (docs).

CHAPTER

FOURTEEN

RESOURCE DISCOVERY

Resource discovery on Chameleon allows you to explore and identify the specific hardware resources available for your experiments. You can discover nodes by their hardware characteristics, view detailed specifications, check maintenance history, and reserve specific resources that meet your experimental requirements.

All physical resources available in Chameleon are described in the Chameleon resource registry. The resource registry is based on the Reference API from the Grid'5000 project. Users can consult the registry via the resource discovery GUI or directly via REST APIs.

Note

This section covers discovery of Chameleon's bare metal resources. CHI@Edge devices are not part of the resource registry and are discovered and managed separately — see the CHI@Edge docs.

Note

Hardware discovery is also available programmatically via the chi.hardware module in python-chi — see our *Jupyter and python-chi guide* for an introduction.

14.1 The hardware catalog on the Chameleon portal

You may use the Hardware Discovery page at the Chameleon Portal to see the different hardware resource types available at each Chameleon site.

14.1.1 Availability calendars

The CHI site buttons in the **Availability Calendar** section of the Resource Browser allow you to open the *The lease calendars* at the Chameleon sites. You must log in using your Chameleon account to view these lease calendars.

14.1.2 Chameleon resource browser

The Resource Browser allows you to **filter Chameleon resources** by node type and view details of each node.

You can filter for specific node features by selecting the checkboxes that match your filter criteria in the menu at the bottom or by clicking on a node type such as *compute_gigaio* (see *Composable Hardware*) or *gpu_a100_pcie*. The numbers printed next to the node types indicate the total number of nodes that we have in our capacity.

You can also click the **Advanced Filters** dropdown to view even more node parameters.

After you have selected filter criteria, you can click the **View** button at the bottom of the page to see details of individual nodes that match your filtering criteria.

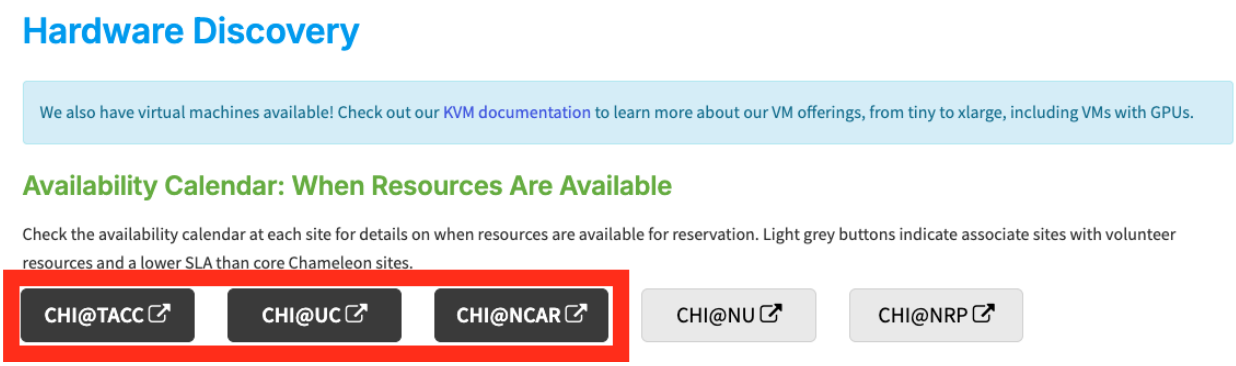


Fig. 1: Resource availability links to the lease calendars

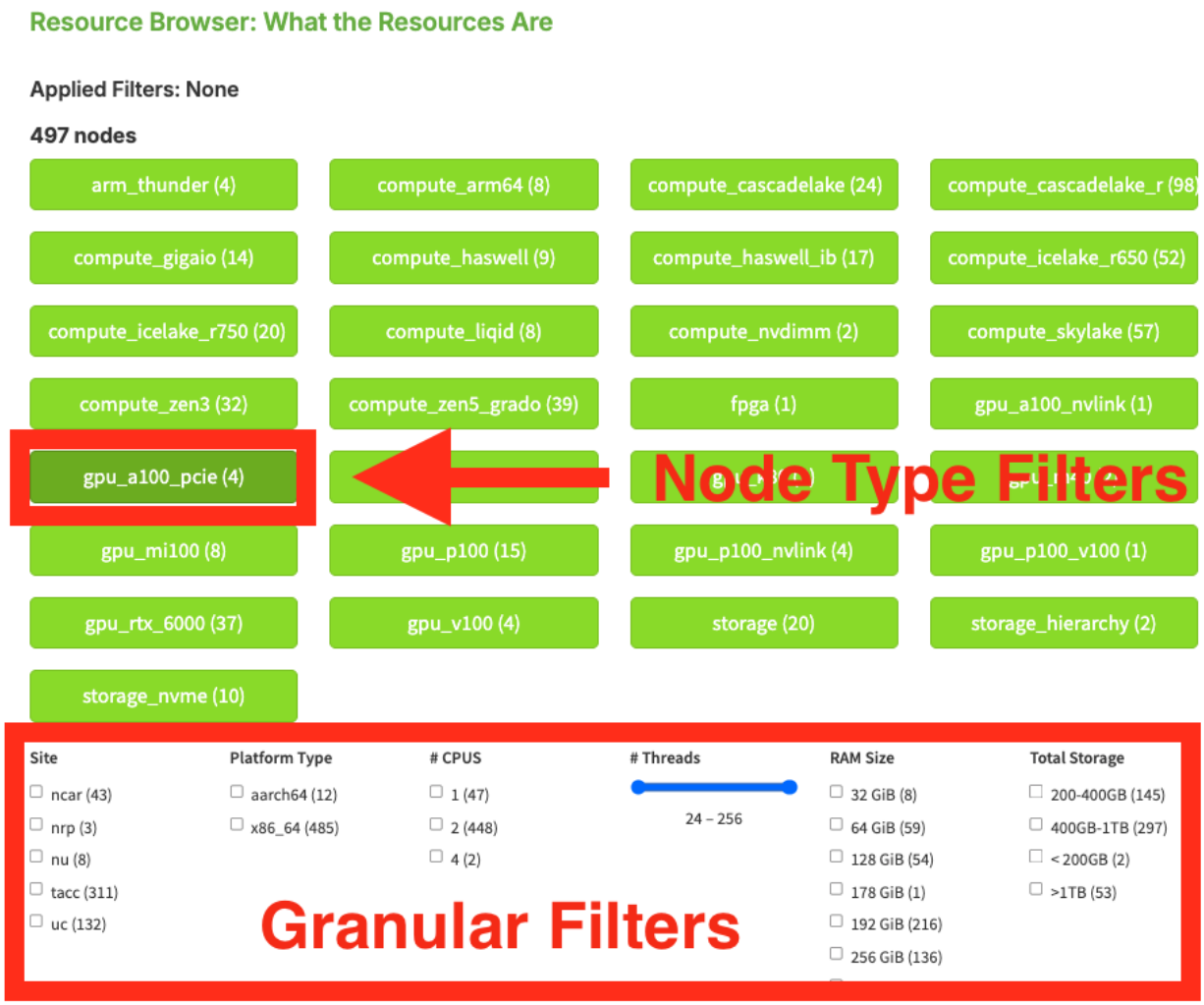
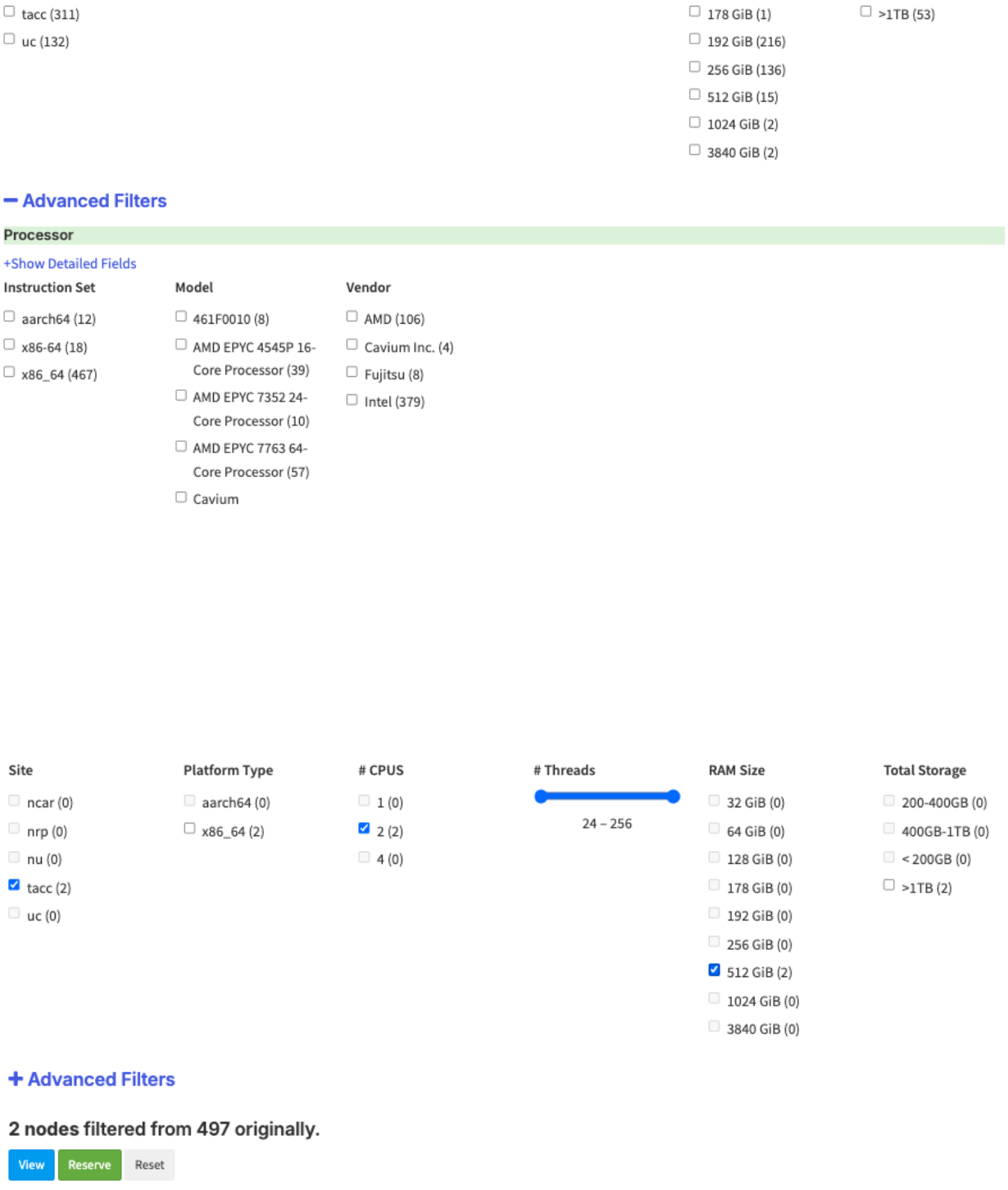


Fig. 2: The Chameleon Resource Browser


tacc (311)
uc (132)
178 GiB (1)
192 GiB (216)
256 GiB (136)
512 GiB (15)
1024 GiB (2)
3840 GiB (2)
>1TB (53)
− Advanced Filters
Processor
+Show Detailed Fields
Instruction Set
aarch64 (12)
x86-64 (18)
x86_64 (467)
Model
461F0010 (8)
AMD EPYC 4545P 16-Core Processor (39)
AMD EPYC 7352 24-Core Processor (10)
AMD EPYC 7763 64-Core Processor (57)
Cavium
Vendor
AMD (106)
Cavium Inc. (4)
Fujitsu (8)
Intel (379)
Site
ncar (0)
nrp (0)
nu (0)
tacc (2)
uc (0)
Platform Type
aarch64 (0)
x86_64 (2)
CPUS
1 (0)
2 (2)
4 (0)
Threads
24 – 256
RAM Size
32 GiB (0)
64 GiB (0)
128 GiB (0)
178 GiB (0)
192 GiB (0)
256 GiB (0)
512 GiB (2)
1024 GiB (0)
3840 GiB (0)
Total Storage
200-400GB (0)
400GB-1TB (0)
< 200GB (0)
>1TB (2)
+ Advanced Filters
2 nodes filtered from 497 originally.
View
Reserve
Reset

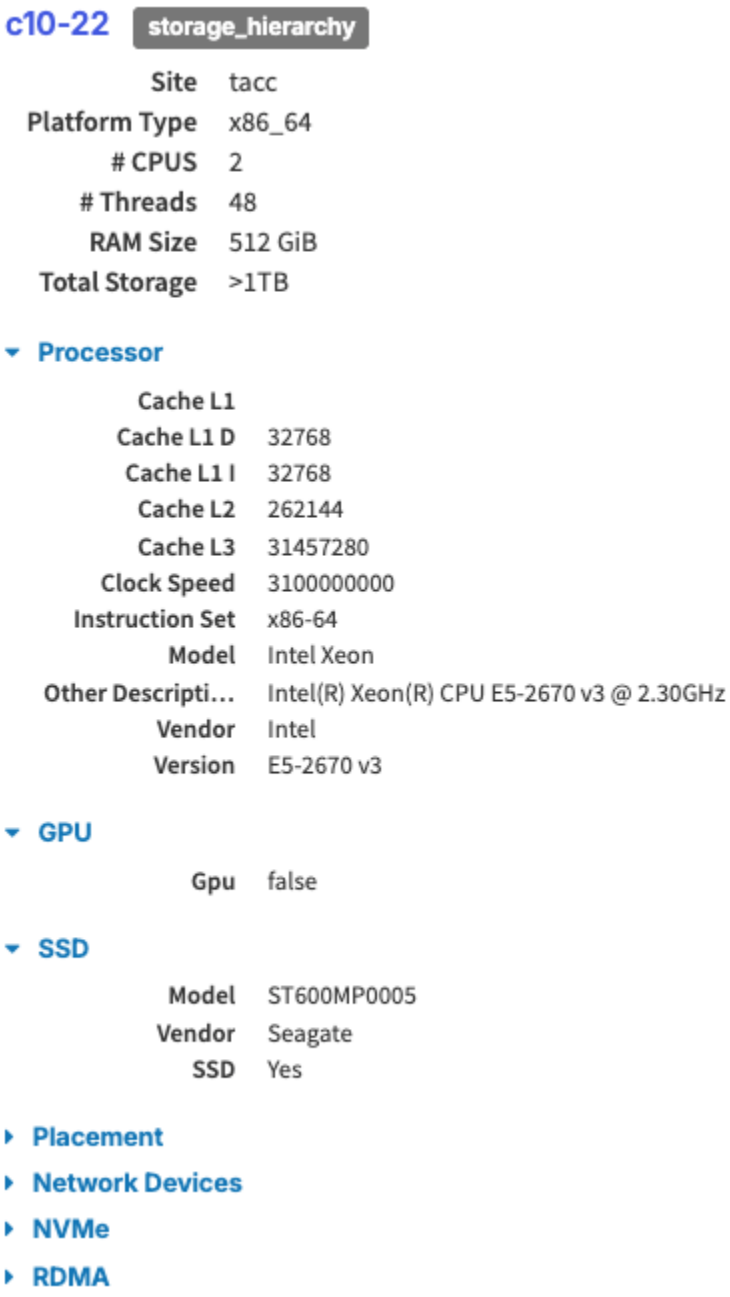


Fig. 3: Node details

Note

All the nodes in Chameleon is identified by their *UUIDs*. You will need the *UUID* of a node for making reservations and for power monitoring. In addition, each node also has a *Version UUID*, which is used for retrieving its maintenance history.

Attention

When we replace faulty hardware on a node, the replacement part typically has the same hardware characteristics. For example, a node with a faulty 250 GB hard drive would be replaced with the same 250 GB hard drive model. However, it may be important for your experimental reproducibility to know about those hardware replacement events, in case it affects your metrics.

14.1.3 Checking availability for node types

From the node detail view, you can click the **View Host Calendar** button to open the site lease calendar for that node. The calendar shows when the node is reserved, letting you identify open windows before making a reservation. For instructions on creating a lease, see *Creating a lease to reserve resources*.

If you have filtered the resource browser down to a single node type, clicking **View Host Calendar** will open the calendar with that node type pre-selected, saving you from having to filter manually after landing on the calendar page. If your current filters produce results spanning multiple node types, the calendar will open in its default view with a default node type loaded instead.

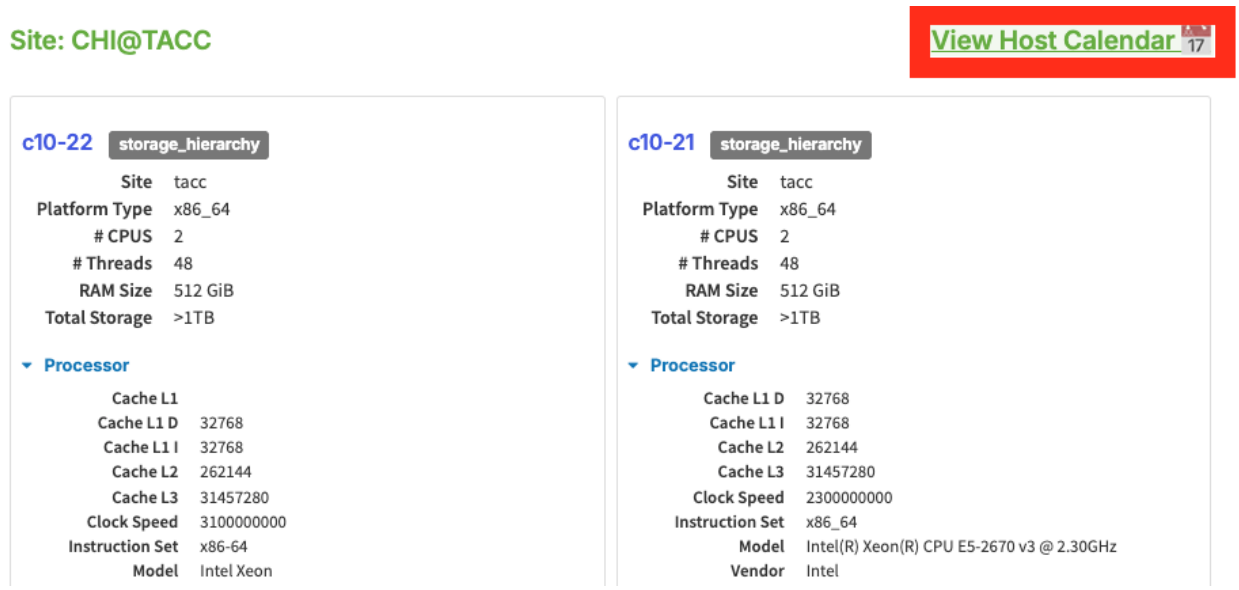
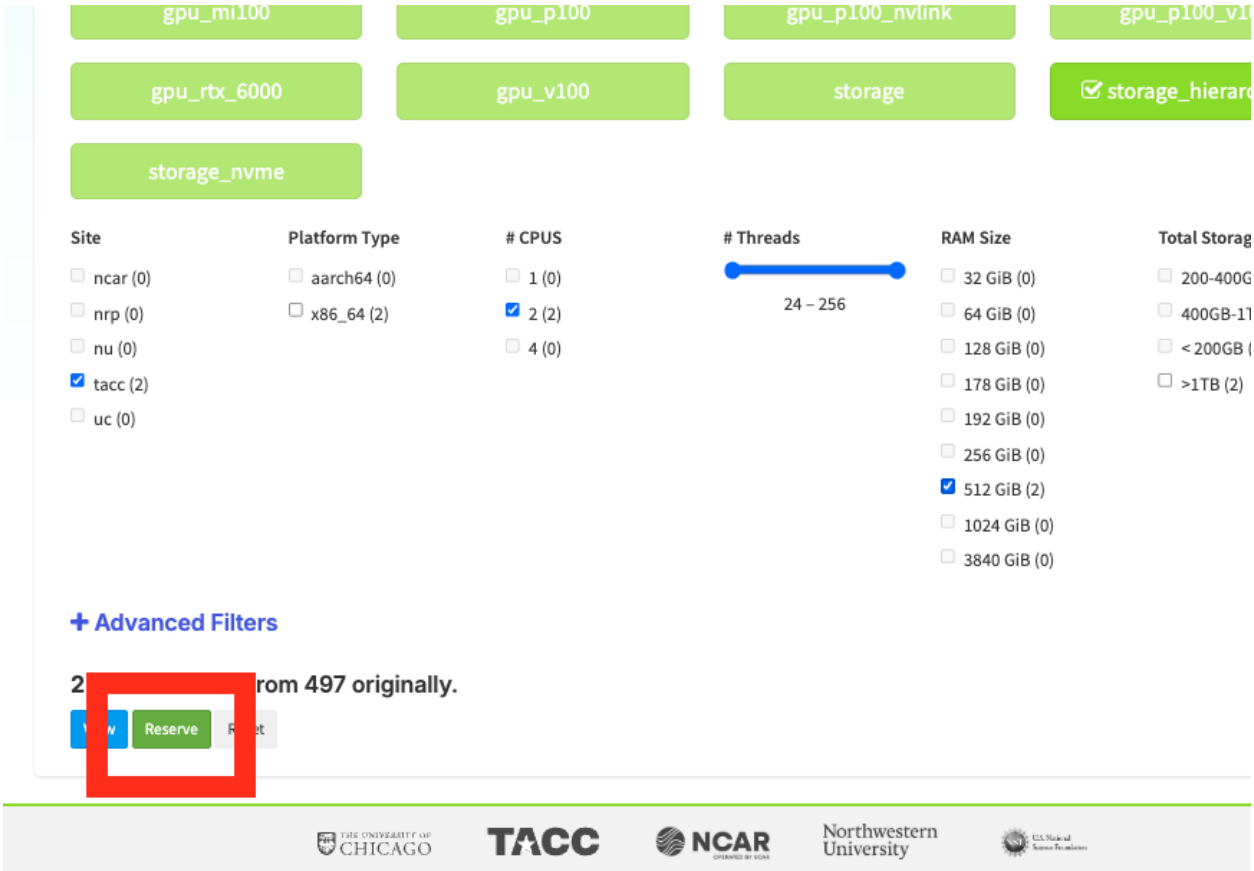


Fig. 4: Node detail view with availability and reserve buttons

Note

You must be logged in to a Chameleon site to view its lease calendar. If you follow a direct link to a site calendar while logged out, you will be redirected to log in and then returned to the calendar automatically.

14.1.4 Generating a reservation script



From the filter page of the Hardware Resource Browser, you can also generate a reservation script by clicking the **Reserve** button at the bottom of the page. The generated command varies depending on your current filter state:

- If you have filtered down to a **single node type**, the command will reserve one node of that type by node type name.
- If your filters produce a list of **nodes across multiple types**, the command will reserve each of those exact nodes by their individual UUIDs.

When clicking the **Reserve** button, a dialog containing the auto-generated command will appear for you to copy and paste.

See also

Streamlining Resource Discovery and Reservations — A Tips&Tricks post walking through the hardware browser's filtering, calendar, and reservation workflow end-to-end with a practical example.

14.2 Using the REST APIs for resource discovery

The Resource Discovery API gives programmatic access to the same hardware inventory that powers the Hardware Discovery page. It is aimed at three main use cases:

- **Scripted node selection** — find nodes matching specific hardware criteria (CPU model, GPU, storage type, etc.) before constructing a reservation.

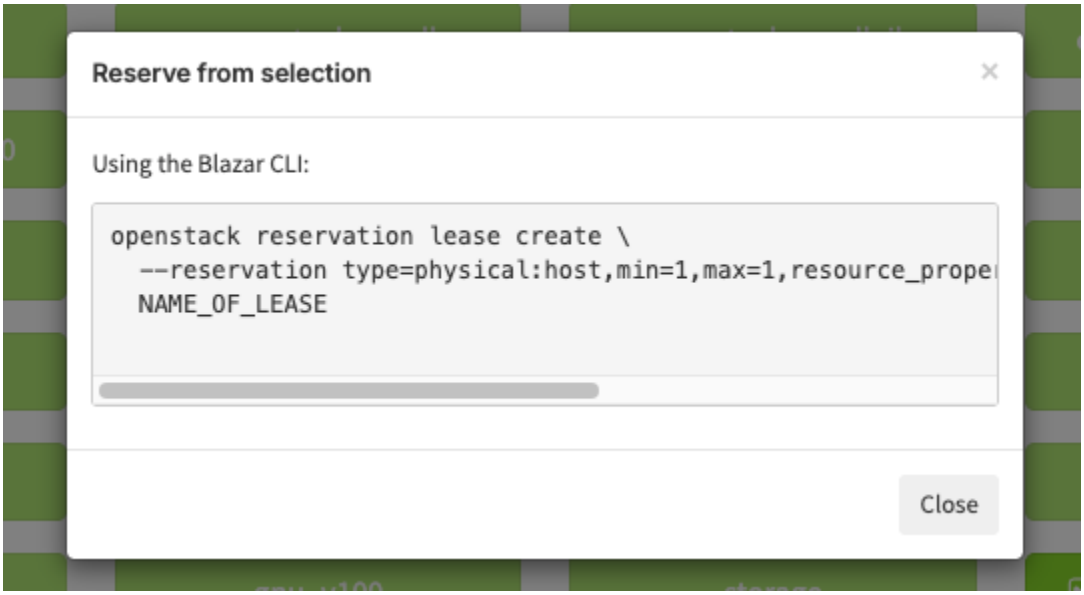


Fig. 5: An auto-generated reservation script

- **Reproducibility tracking** — detect hardware changes between experiment runs using the versioned change history.
- **Tool building** — integrate Chameleon hardware data into your own dashboards or automation frameworks.

For most users, the Hardware Discovery page or the *python-chi* library is the easier path. This API is for power users and integrators who need direct, scriptable access.

The API uses a REST architecture over HTTP, so any HTTP client works: cURL, a browser, or any language's HTTP library. It also implements "Hypermedia as the Engine of Application State" (HATEOAS) — every response includes `links` that point to related resources, so you can traverse the full hierarchy by following links rather than constructing URLs manually.

14.2.1 Prerequisites

Chameleon uses cURL to interact with the API. The User-Agent cURL is a command line tool for transferring data with URL syntax, supporting many protocols including HTTP and HTTPS.

To install cURL, follow the instructions below:

OS X

cURL is installed by default on OS X. Nothing to do for you!

Linux

Use your package manager to install cURL. Either (Debian/Ubuntu-based distributions):

```
$ sudo apt-get install curl
```

or (RedHat-based distributions):

```
$ sudo yum install curl
```

Windows

Download and install the cURL package from the website.

14.2.2 Your first requests

The API entry-point for the resource discovery API is located at https://api.chameleoncloud.org/. Open your Terminal program (or the cURL executable if you're on Windows), and use cURL to fetch the resource located at that URL:

```
curl -i https://api.chameleoncloud.org/
```

Tip

The `-i` flag tells cURL to display the HTTP header in addition to the HTTP body.

Below is what you should see in response:

```
HTTP/2 200
date: Fri, 01 May 2026 02:17:01 GMT
content-type: application/json
content-length: 250
strict-transport-security: max-age=15724800; includeSubDomains

{"type":"grid","uid":"chameleoncloud","version":"aaa09ab330838062ed66ee8a3841e90fe9495039
↪","timestamp":"1776354589","links":[{"rel":"sites","href":"/sites"},{"rel":"self","href
↪":"/"},{"rel":"parent","href":"/"},{"rel":"versions","href":"/versions"}]}
```

Note

The HTTP status of `200 OK` indicates that the server is able to process your request and that everything is fine.

Tip

Pipe any response through `jq` to pretty-print it and make it easier to read:

```
curl https://api.chameleoncloud.org/ | jq
```

You may notice that the response contains a number of link elements, which advertise other resources that you can access. For example, let's fetch the `/sites` resource.

```
curl https://api.chameleoncloud.org/sites | jq
```

The response should look like:

Note

The actual response includes all 6 Chameleon sites. Two are shown here for brevity.

```
{
  "total": 6,
  "offset": 0,
  "items": [
    {
```

(continues on next page)

(continued from previous page)


```
    "uid": "tacc",
    "name": "CHI@TACC",
    "description": "Texas Advanced Computing Center",
    "email_contact": "help@chameleoncloud.org",
    "latitude": 30.390223,
    "longitude": -97.72563,
    "location": "Austin, Texas, USA",
    "security_contact": "help@chameleoncloud.org",
    "site_class": "baremetal",
    "sys_admin_contact": "help@chameleoncloud.org",
    "user_support_contact": "help@chameleoncloud.org",
    "web": "https://chi.tacc.chameleoncloud.org",
    "version": "aaa09ab330838062ed66ee8a3841e90fe9495039",
    "links": [
      {
        "rel": "self",
        "href": "/sites/tacc"
      },
      {
        "rel": "parent",
        "href": "/"
      },
      {
        "rel": "clusters",
        "href": "/sites/tacc/clusters"
      },
      {
        "rel": "versions",
        "href": "/sites/tacc/versions"
      },
      {
        "rel": "version",
        "href": "/sites/tacc/versions/aaa09ab330838062ed66ee8a3841e90fe9495039"
      }
    ],
    "type": "site"
  },
  {
    "uid": "uc",
    "name": "CHI@UC",
    "description": "University of Chicago",
    "email_contact": "help@chameleoncloud.org",
    "latitude": 41.718002,
    "longitude": -87.982952,
    "location": "Argonne National Laboratory, Lemont, Illinois, USA",
    "security_contact": "help@chameleoncloud.org",
    "site_class": "baremetal",
    "sys_admin_contact": "help@chameleoncloud.org",
    "user_support_contact": "help@chameleoncloud.org",
    "web": "https://chi.uc.chameleoncloud.org",
    "version": "aaa09ab330838062ed66ee8a3841e90fe9495039",
    "links": [
```


(continues on next page)

(continued from previous page)


```
        {
          "rel": "self",
          "href": "/sites/uc"
        },
        {
          "rel": "parent",
          "href": "/"
        },
        {
          "rel": "clusters",
          "href": "/sites/uc/clusters"
        },
        {
          "rel": "versions",
          "href": "/sites/uc/versions"
        },
        {
          "rel": "version",
          "href": "/sites/uc/versions/aaa09ab330838062ed66ee8a3841e90fe9495039"
        }
      ],
      "type": "site"
    }
  ],
  "version": "aaa09ab330838062ed66ee8a3841e90fe9495039",
  "links": [
    {
      "rel": "self",
      "href": "/sites"
    },
    {
      "rel": "parent",
      "href": "/"
    }
  ]
}
```


Note

Previous versions of the API included a `"type"` field (e.g. `"type": "application/vnd.grid5000.collection+json"`) on every link object in responses. This field has since been removed and no longer appears in API responses.

14.2.3 Discover resources

It is easy to discover resources using REST APIs when you chase down the `links` in the responses.

As seen in the previous section, when you fetch the API root resource, you can find the link to the collection of sites. If you look at the site description, you will find a list of links to other resources. For example, each site has a link named `clusters`. When you fetch this link, it returns the list of clusters on that site.

> **Note**
>
> In this API, a "cluster" is a logical grouping of nodes at a site inherited from the Grid'5000 data model — it is not an HPC-style compute cluster. Every baremetal site (CHI@TACC, CHI@UC, CHI@NCAR, CHI@NU, CHI@NRP) has exactly one cluster, always named `chameleon`. CHI@Edge has no clusters and its devices are not accessible through this API path.
>
> In practice this means the `clusters` level is a fixed pass-through: the path to nodes at any baremetal site is always `/sites/{site}/clusters/chameleon/nodes`.

For example, to get clusters at *CHI@TACC*:

```
curl https://api.chameleoncloud.org/sites/tacc/clusters | jq
```

There is a link named `nodes` in each cluster description, which returns the list of nodes for that cluster.

> **Tip**
>
> Since every baremetal site uses the same cluster name, you can go directly to the nodes endpoint without traversing the clusters step:
>
> ```
> curl https://api.chameleoncloud.org/sites/tacc/clusters/chameleon/nodes | jq
> ```

You should get back a large collection of nodes. Each node is described in detail, so you can programmatically find the nodes most suitable for your experiments.

The following command examples allow you to see that some of the nodes on the *chameleon* cluster at *TACC* have a different disk configuration:

```
curl https://api.chameleoncloud.org/sites/tacc/clusters/chameleon/nodes/f503a229-9d71-
↪4819-bf56-5d8490b29e7c | jq | grep -A 10 storage_devices
curl https://api.chameleoncloud.org/sites/tacc/clusters/chameleon/nodes/d4a46dc6-7cac-
↪417f-800c-faea63a46130 | jq | grep -A 10 storage_devices
```

14.2.4 Fetch the latest changes

Chameleon hardware is added, updated, or removed over time — including component replacements that may affect your results even when a node's overall specifications appear unchanged. The versioned change history lets you detect these events, which is useful for experimental reproducibility.

To fetch the change history for a site:

```
curl https://api.chameleoncloud.org/sites/tacc/versions | jq
```

Each version in the response represents a change to some resource at that site. You can compare versions across experiment runs to verify that the hardware environment was consistent.

CHAPTER

FIFTEEN

RESERVATIONS

Unlike virtual resources on a regular on-demand cloud, physical resources on Chameleon must be reserved before using them for an experiment. Once a reservation has been accepted, users are guaranteed that resources will be available at the time they chose (except in extraordinary circumstances such as hardware or platform failures), which helps to plan large scale experiments.

Chameleon resources are reserved via Blazar which provides Reservation as a Service for OpenStack.

Three types of resources can be reserved: physical bare metal hosts, network segments (VLANs), and floating IPs.

Note

This section covers Blazar leases for bare metal resources. CHI@Edge uses a different, container-lease model rather than Blazar reservations — see the CHI@Edge docs.

Note

Leases can also be created and managed programmatically via the chi.lease module in python-chi — see our *Jupyter and python-chi guide* for an introduction.

Attention

A note on lease stacking

To prevent resource hoarding and ensure fair access to specialized hardware, Chameleon discourages "lease stacking" or making multiple overlapping reservations. Review our lease stacking policy to align your reservations with community practices for efficient resource use.

15.1 Provisioning and managing resources using the GUI

To make reservations of the resources, first log into the Horizon GUI - either CHI@TACC or CHI@UC. Then, choose a project and configure your local timezone. For details on how to choose a project and update personalized settings, see *Graphical User Interface (GUI)*.

In the navigation sidebar, go to the *Reservations* section and click *Leases*.

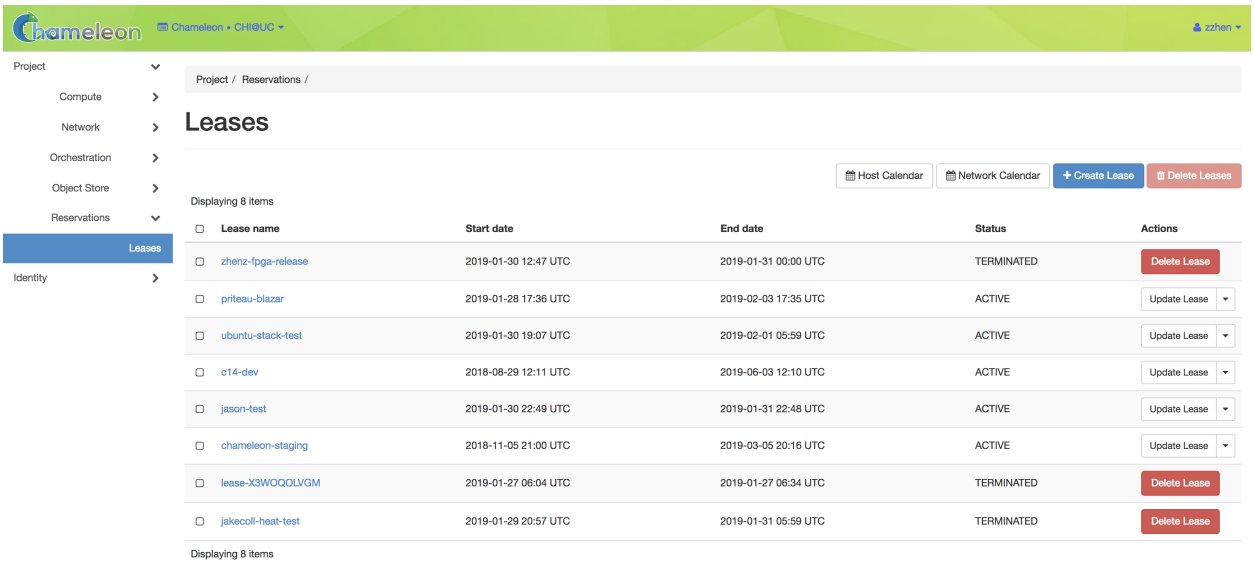


Fig. 1: The Leases page in the GUI

15.1.1 The lease calendars

To discover when resources are available, You can access the lease calendars by clicking on the *Host Calendar* button for physical hosts and clicking on the *Network Calendar* button for VLANs. This will display a Gantt chart of the reservations which allows you to find when resources are available. The *Y* axis represents the different physical nodes in the system and the *X* axis represents time.

> **Tip**
>
> Education projects may require "sub-leases" to facillitate student access to a node during a project, while keeping that node available to the project. This ensures that resources required for a class are not unavailable before a deadline. If this is required for your usage, we can temporarily grant exclusive access to a node to your project. Create a lease for the node, and contact the Help Desk to request exclusive node access for your project.

> **Tip**
>
> The nodes and VLANs are identified by their *UUIDs*. The colors are used to indicate different reservations, i.e. the resources that belong to the same reservation are colored the same. Hovering over the chart provides the details about the reservation. To change the display time frame, click on `1d`, `1w`, and `1m` buttons or fill in the start and end times.

15.1.2 Creating a lease to reserve resources

Once you have chosen a time period when you want to reserve resources, go back to the *Leases* screen and click on the *Create Lease* button. It should bring up the window displayed below:

1. Pick a name for the lease. The name needs to be unique across your project.
2. Pick a start time and lease duration in days. If you would like to start your lease as soon as possible, you may leave the start time blank and Chameleon will attempt to reserve your nodes to begin immediately with a default Lease duration of 1 day.

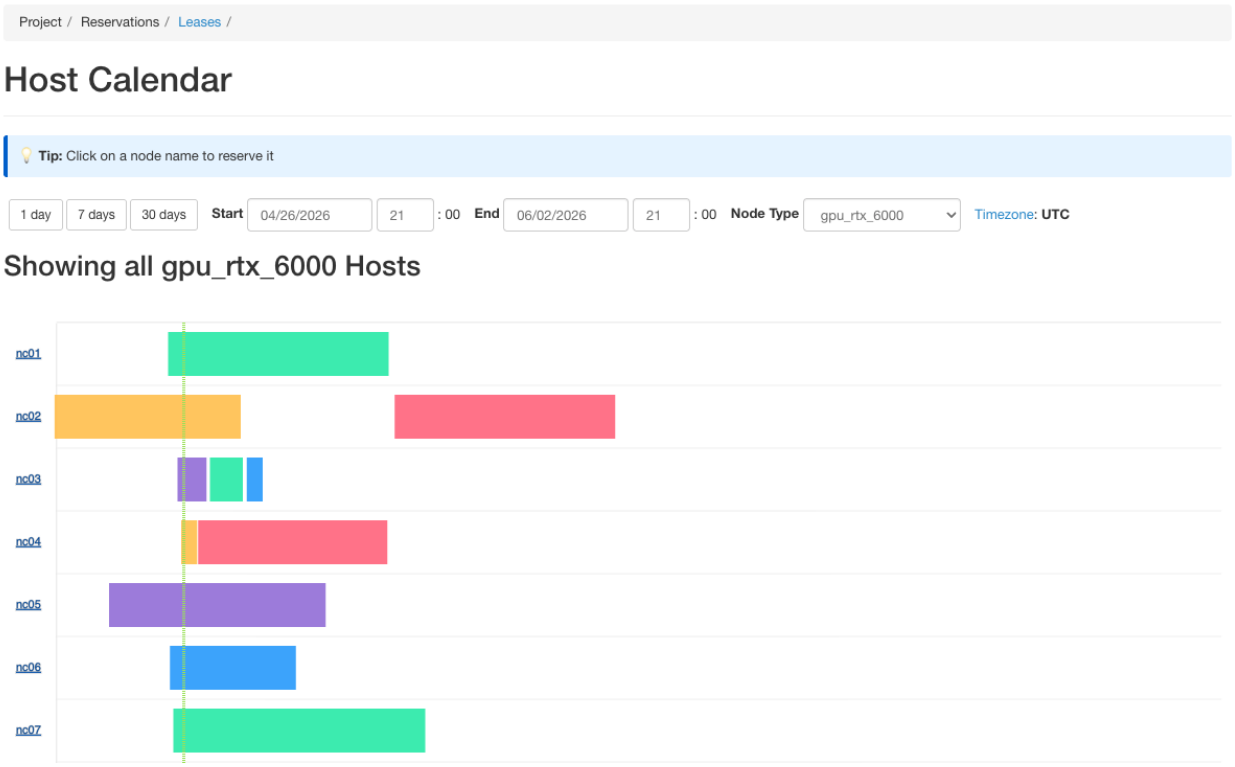


Fig. 2: The Host Calendar

> **Note**
>
> If you have not selected a timezone earlier, the default timezone is **UTC**. Therefore, the date must be entered in **UTC**!

> **Tip**
>
> You can get the UTC time by running `date -u` in your terminal.

3. To reserve a bare metal node, navigate to the "Hosts" tab.

    a. Check "Reserve Hosts".

    b. Choose the minimum and maximum number of hosts.

    c. Choose a node type in the drop down menu below the *node_type* and = drop down lists.

    > **Note**
    >
    > You may only request one type of node in each individual lease. If you wish to request multiple node types, you must create separate Leases for each node type.

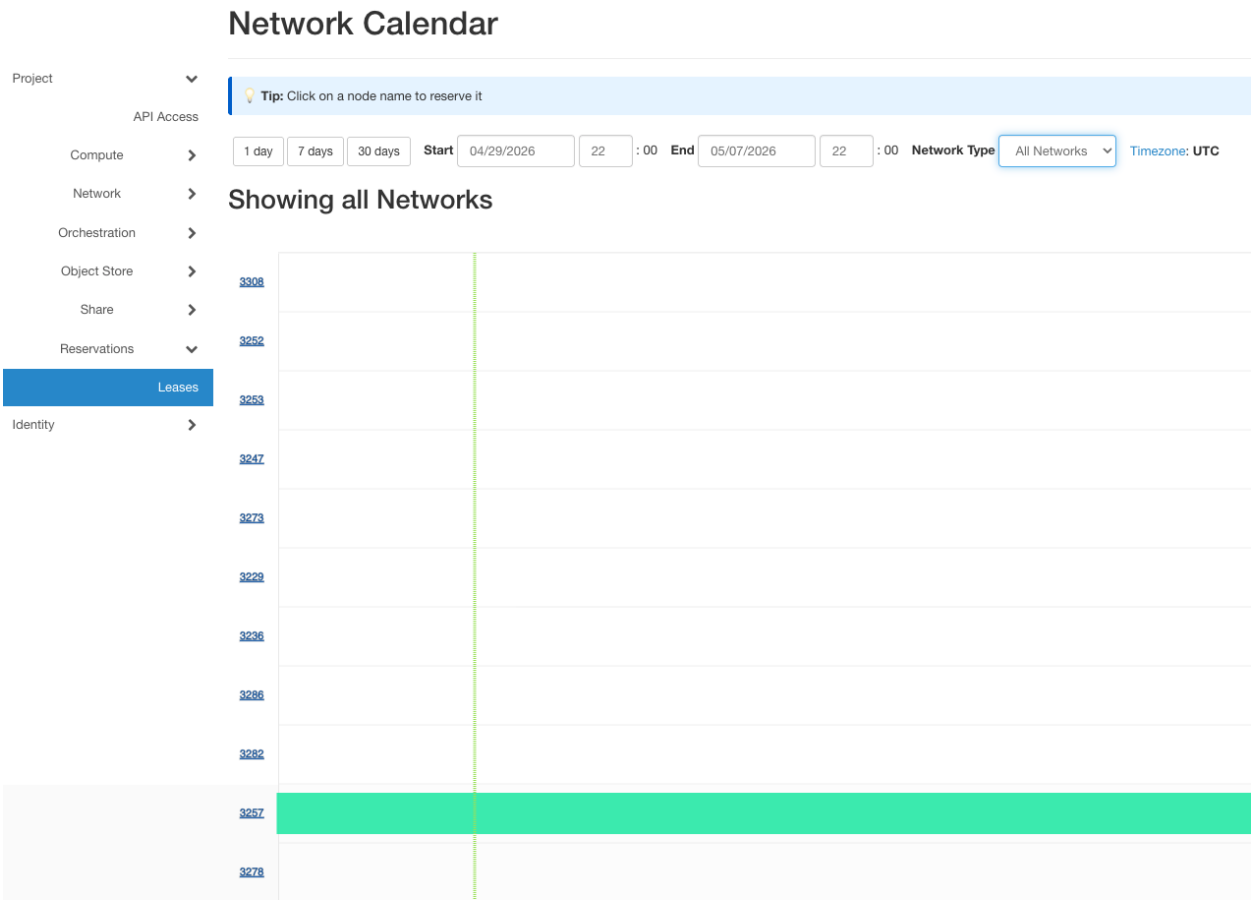


Fig. 3: The Network Calendar

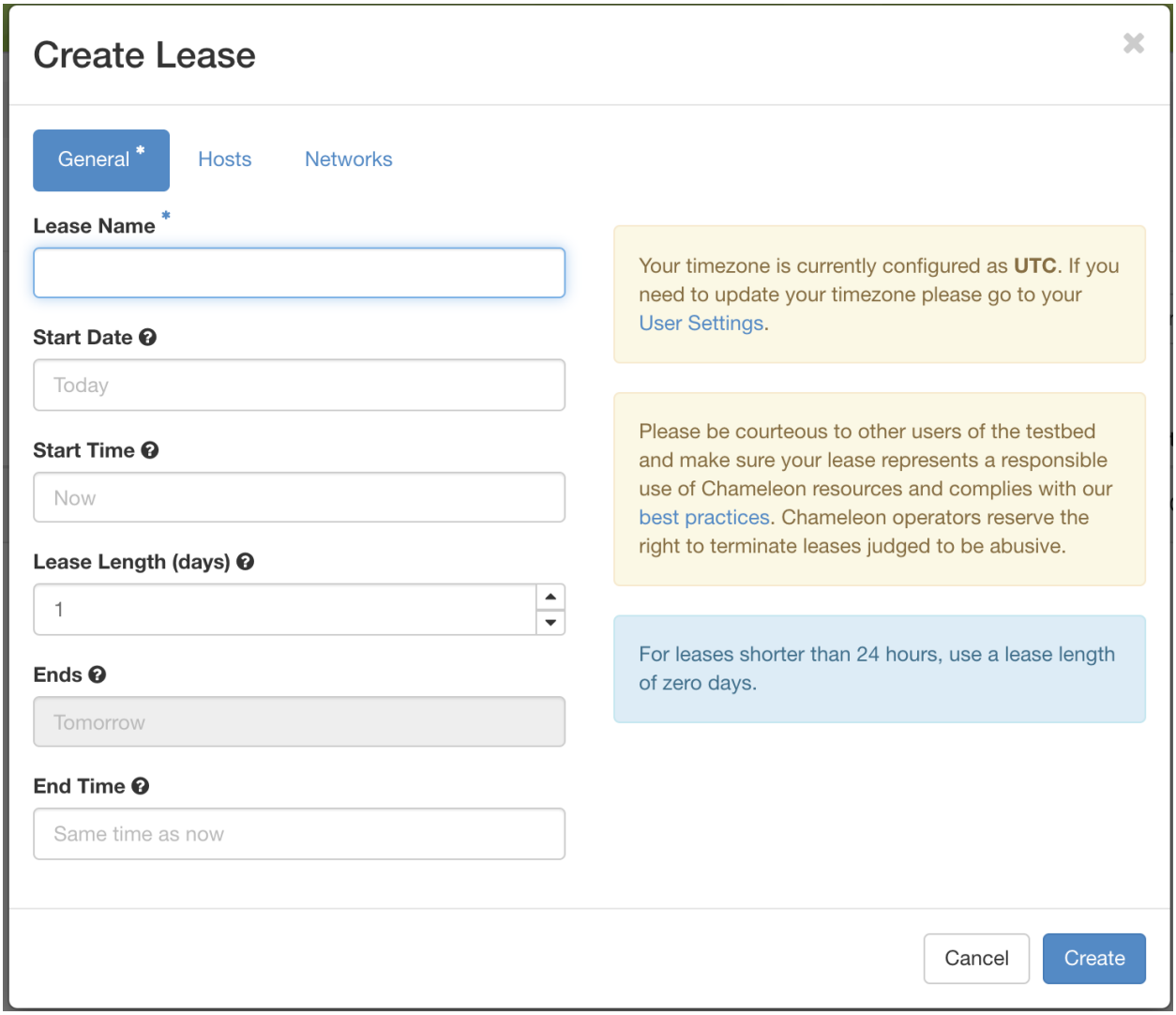


Fig. 4: The Create Lease dialog

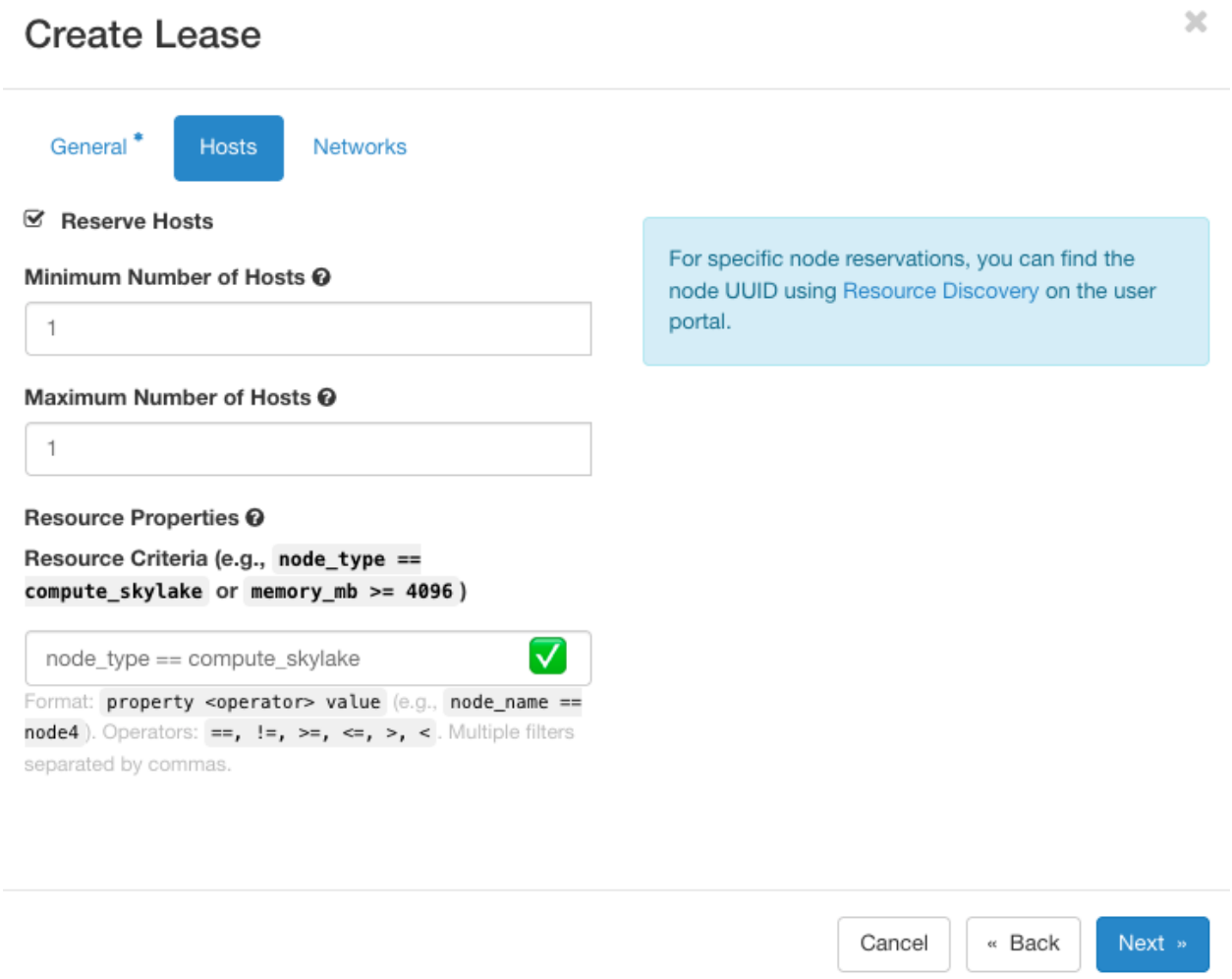


Fig. 5: The Create Lease dialog Host reservation tab

> **Warning**
>
> You must select = when matching the node_type to a specific selection. Using other operators like >= may yield unexpected results.

4. To reserve a vlan segment, navigate to the “Networks” tab.

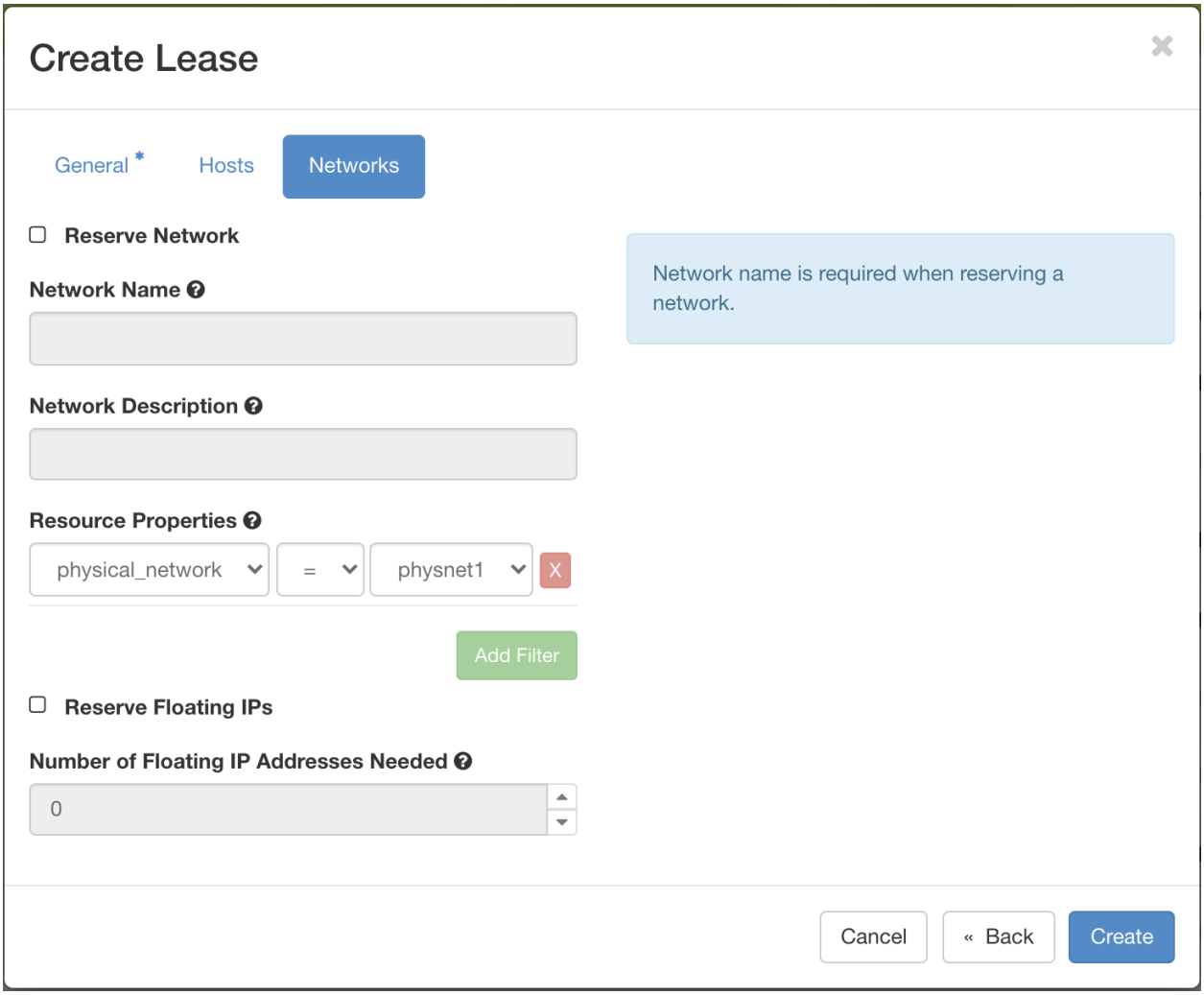


Fig. 6: The Create Lease dialog Network reservation tab

a. Check “Reserve Network”

b. Enter the network name and description

> **Note**
>
> When a VLAN segment reservation ends, all Neutron resources attached to the network will be automatically deleted. Bare metal instances using the network will lose network connectivity.

> **Tip**
>
> Network name is required when reserving VLAN segment.

5. To reserve floating IPs, navigate to the "Networks" tab.
    a. Check "Reserve Floating IPs".
    b. Choose the number of floating IPs.
6. Click on the *Create* button.

Once created, the lease details will be displayed. At the bottom of the page are the details about the reservation. Initially the reservation is in the `Pending` status, and stays in this state until it reaches the start time.

> **Tip**
>
> If you want Blazar to launch an instances or complex appliance as soon as the lease starts, read the `Advanced Reservation Orchestration` section our *Complex Appliances* documentation.

Once the start time of the lease is reached, the lease will be started and its reservation will change to `Active`; you may need to refresh the page to see the updates.

> **Tip**
>
> The lease is identified by a *UUID*. You may find it useful when using the CLI or submitting tickets on our Help Desk.

> **Attention**
>
> To ensure fairness to all users, resource reservations (leases) are limited to a duration of 7 days. However, an active lease within 48 hours of its end time can be prolonged by up to 7 days from the moment of request if resources are available.
>
> Chameleon will send an email reminder to you 48 hours before your lease ends. If your lease duration is less than 48 hours, Chameleon will send you an email right after your lease is created. You can *disable the email notification by using the command line*.

> **Note**
>
> This 7-day limit applies to bare metal host leases. KVM@TACC instance leases follow a different policy: 6 months for standard flavors, but the same 7-day cap for GPU-attached flavors. See the Chameleon FAQ for the authoritative policy.

15.1.3 Extending a lease

To prolong a lease, click on the *Update Lease* button in *Actions* column.

Fill out the form by specifying the amount of additional time to add to the lease. Then, click on the *Update* button to finish your request.

Lease Detail

Lease

Name	mike-terraform
Id	6e790660-a692-473f-9429-eeeac1ae7339
Project Id	f6c7696906c04b3c89fc3bda9a1b8be0
Start date	2023-07-24 20:43 UTC
End date	2023-07-25 20:42 UTC
Status	ACTIVE
Degraded	No

Events

end_lease	• *Status:* Undone • *Time:* 2023-07-25 20:42 UTC
before_end_lease	• *Status:* Done • *Time:* 2023-07-24 20:43 UTC
start_lease	• *Status:* Done • *Time:* 2023-07-24 20:43 UTC

Reservations

id	90d25f80-f25f-44b5-8168-d0767c5c0bba
status	active
resource type	physical:host
missing resources	No
resources changed	No
hypervisor_properties	-
resource_properties	-
before_end	default
on_start	default
min	1
max	1

Nodes

c01-39 (uid: 83830c1b-391b-42cb-8c2b-37d251d55ae5) **(status: healthy)** Re-Allocate Host

Fig. 7: Lease details page

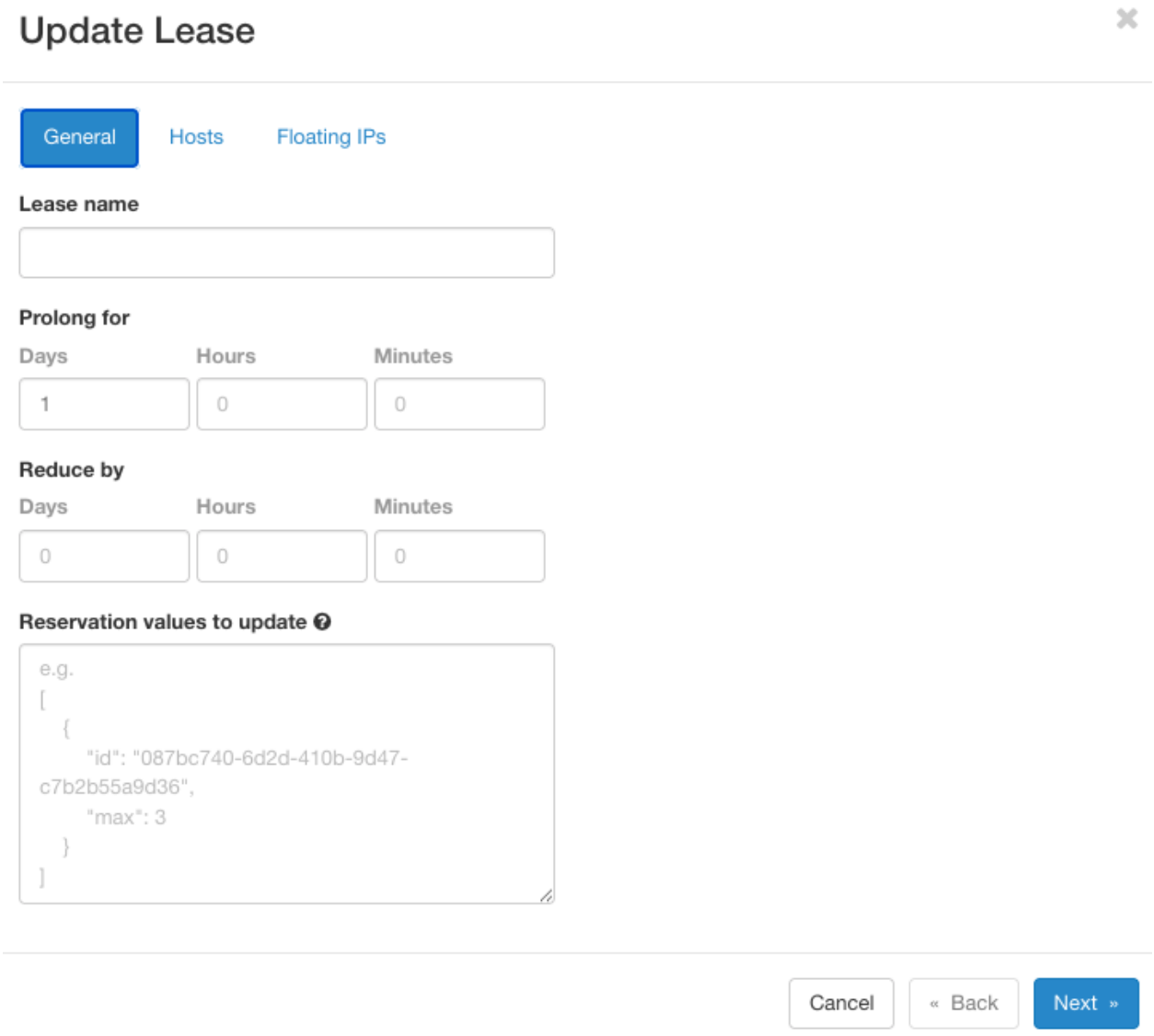


Fig. 8: The Update Lease Parameters dialog

Tip

If there is an advance reservation blocking your lease prolongation that could potentially be moved, you can interact through the users mailing list to coordinate with others users. Additionally, if you know from the start that your lease will require longer than a week and can justify it, you can submit a ticket on our Help Desk to request a **one-time exception** of creating a longer lease. You may also other exceptions to our other policies, such as idle lease termination, by submitting a request.

Changing the number of nodes of a lease

It is now possible to change the number of nodes reserved in a lease. For advance reservations that haven't yet started, the node count can be increased or decreased. For reservations already started, only new nodes can be added.

To change the number of nodes of a lease, click on the *Update Lease* button in *Actions* column.

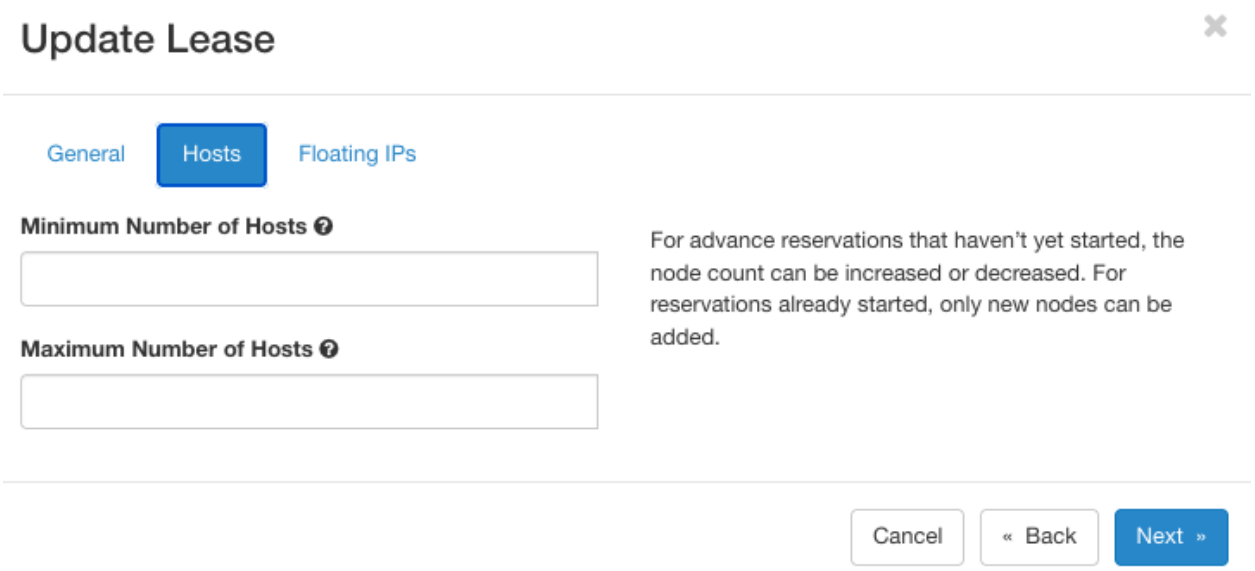


Fig. 9: The Update Lease Parameters dialog, changing the number of reserved nodes

Navigate to the "Hosts" tab, and fill out the form by specifying the new minimum and maximum numbers of hosts. Then, click on the *Update* button to finish your request.

Changing the number of floating IPs in a lease

It is possible to change the number of floating IPs in a lease, whether the lease is pending or active. In some situations, you cannot renew a lease due to another user reserving the same floating IP in your lease. In this case, you can set your lease to have 0 floating IPs, and create a second lease just for reserving floating IPs.

To change the number of floating IPs, click on the *Update Lease* button in *Actions* column.

Navigate to the "Floating IPs" tab, and fill out the form by specifying the amount of floating IPs. Then, click on the *Update* button to finish your request.

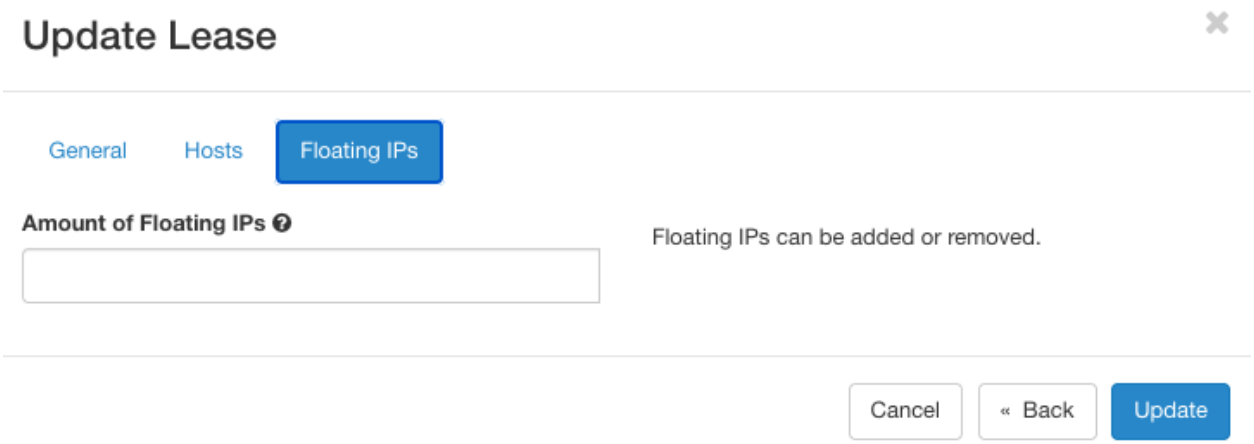


Fig. 10: The Update Lease Parameters dialog, changing the number of reserved IPs

15.1.4 Reserving a node by UUID

You may reserve a specific node by providing its *UUID*. To learn more about how to find a node with a specific type, see *Resource Discovery*. In the *Create Lease* dialog, use the **Resource Properties** bar to filter by the *uid* keyword then paste the ID of your desired node.

15.2 Provisioning and managing resources using the CLI

The sections above present the most user friendly mode of usage, with most actions performed via the GUI. However, Chameleon can be accessed via the OpenStack command line tools which provides more capabilities. This section presents some advanced usage using the command line tools.

Tip

Reading *Command Line Interface (CLI)* is highly recommended before continuing on the following sections.

15.2.1 Blazar client installation

To reserve specific nodes, based on their identifier or their resource specifications, you must use the Blazar command line client. To use the CLI, you must install the `python-blazarclient`. To install `python-blazarclient`, run the following command:

```
pip install git+https://github.com/ChameleonCloud/python-blazarclient.git@chameleoncloud/
↪xena
```

Note

To reserve VLAN segments or floating IPs, you must use a Chameleon fork of the Blazar client, as above.

Before using *Blazar Client*, You must configure the environment variables for your project via `source` *the OpenStack RC Script* or use the CLI switches every time you run the commands. Type `blazar` in your terminal session to enter the *Interactive Mode*. You may also use `blazar` in the *Shell Mode*.

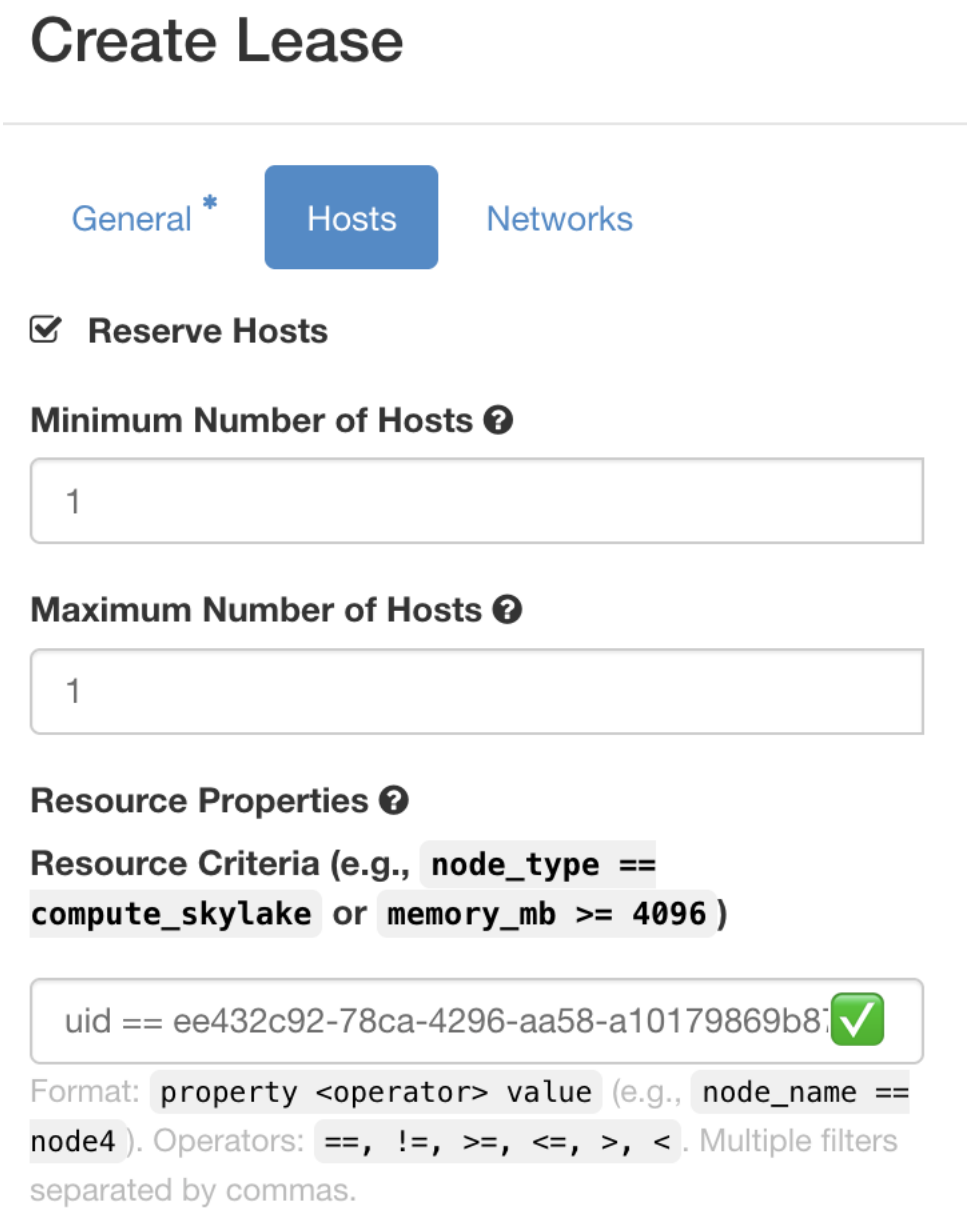


Fig. 11: Selecting a node by UUID

Warning

Two common causes of an `Internal Server Error` when creating a lease:

- Your application credential does not have the **Unrestricted** checkbox enabled — see *Creating an application credential*. Blazar requires this to create reservations.
- Using == instead of = as the comparison operator inside `resource_properties`. The query language used by `resource_properties` takes a single = for equality, as in the examples below — == is not valid and will cause the request to fail.

15.2.2 Creating a lease to reserve physical hosts

To create a lease, use the `lease-create` command. The following arguments are required:

- `--reservation` with the `min`, `max`, `resource_type`, and `resource_properties` attributes
- `--start-date` in `"YYYY-MM-DD HH:MM"` format
- `--end-date` in `"YYYY-MM-DD HH:MM"` format
- A lease name

If `--start-date` is ommitted, then the current date and time will be used by default.

For example, the following command will create a lease with the name of `my-first-lease` and the node type of `compute_skylake` that starts on June 17th, 2022 at 4:00pm and ends on June 17th, 2022 at 6:00pm:

```
openstack reservation lease create \
  --reservation min=1,max=1,resource_type=physical:host,resource_properties='["=", "
↪$node_type", "compute_skylake"]' \
  --start-date "2022-06-17 16:00" \
  --end-date "2022-06-17 18:00" \
  my-first-lease
```

Instead of specifying the node type, you may also reserve a specific node by providing it's *UUID*. For example, to reserve the node with *UUID* of `c9f98cc9-25e9-424e-8a89-002989054ec2`, you may run the command similar to the following:

```
openstack reservation lease create \
  --reservation min=1,max=1,resource_type=physical:host,resource_properties='["=", "$uid
↪", "c9f98cc9-25e9-424e-8a89-002989054ec2"]' \
  --start-date "2022-06-17 16:00" \
  --end-date "2022-06-17 18:00" \
  my-custom-lease
```

To create a lease with multiple resource properties, you must combine them like `["and", [property1], [property2], [...] ]`. For example, to reserve a node with *$architecture.smt_size* of *48* and *node_type* of *compute_haswell*:

```
openstack reservation lease create \
  --reservation min=1,max=1,resource_type=physical:host,resource_properties='["and", ["=
↪", "$architecture.smt_size", "48"], ["=", "$node_type", "compute_haswell"]]' \
  --start-date "2022-06-17 16:00" \
  --end-date "2022-06-17 18:00" \
  my-custom-lease
```

Attention

To specify a `before_end` action, simply add `before_end=<action_type>` to `reservation` parameter. For example:

```
openstack reservation lease create \
  --reservation min=1,max=1,resource_type=physical:host,resource_properties='["=", "
↪$uid", "c9f98cc9-25e9-424e-8a89-002989054ec2"]',before_end=email \
  --start-date "2022-06-17 16:00" \
  --end-date "2022-06-17 18:00" \
  my-custom-lease
```

Currently supported `before_end` action types include

Action Type	Description
`email`	Send an email notification.
`default`	Default action used when no action is specified; Currently set to `email`.
`''`	Do nothing.

The default `before_end` action is set to `email`. To disable the email notification, set `before_end=''`.

Actually, you may use any resource property that is in the resource registry to reserve the nodes. To see the list of properties of nodes, first get the full list of nodes with the command:

```
openstack reservation host list
```

The output should look like:

```
+------+--------------------------------------+-------+-----------+----------+
| id   | hypervisor_hostname                  | vcpus | memory_mb | local_gb |
+------+--------------------------------------+-------+-----------+----------+
| 151  | 00401ba8-4fb0-4f1e-a7dc-e93065ebdd15 |    24 |    128000 |      200 |
| 233  | 004c89fa-ff13-4563-9012-f2d62c1a7aff |    24 |    128000 |      200 |
| 330  | 01029fb8-0a0b-4949-92b0-a756fb8588e5 |    24 |    128000 |      200 |
| 146  | 036b16e3-9fa6-442c-8e6d-cfe12ed5c8a3 |    24 |    128000 |      200 |
| 992  | 05dd5e25-440f-4492-b3b8-9d39af83b8bc |     8 |      3200 |      100 |
| 219  | 066d92f5-7cb9-49ea-8f05-842566672ebf |    24 |    128000 |      200 |
| 3216 | 06b164d5-3514-4ebe-8928-0bd2f9508b80 |     0 |         0 |        0 |
| 156  | 07030786-d6e8-46b4-b0f2-79b0b303b518 |    24 |    128000 |      200 |
| 212  | 07051549-c404-44af-8e73-8beb5891864a |    24 |    128000 |      200 |
| 175  | 07fd65f0-b814-429b-a2fb-3a4afa52de41 |    24 |    128000 |      200 |
| 255  | 081d2cb1-b6b5-4014-b226-7a42d8588307 |    24 |    128000 |      200 |
```

To get resource properties of a host, run `host-show` command with the `id` listed in the first column. For example, to get the resource properties of the host 151, run:

```
openstack reservation host show 151
```

The output should look like:

```
+--------------------------------+--------------------------------------------+
| Field                          | Value                                      |
+--------------------------------+--------------------------------------------+
```

(continues on next page)

(continued from previous page)

```
| architecture.platform_type       | x86_64                                          |
| architecture.smp_size            | 2                                               |
| architecture.smt_size            | 48                                              |
| bios.release_date                | 03/09/2022                                      |
| bios.vendor                      | Dell Inc.                                       |
| bios.version                     | 1.2                                             |
| chassis.manufacturer             | Dell Inc.                                       |
| chassis.name                     | PowerEdge R630                                  |
| chassis.serial                   | 4VJGD42                                         |
| cpu_info                         | baremetal cpu                                   |
| created_at                       | 2022-06-26 20:50:58                             |
| gpu.gpu                          | False                                           |
| hypervisor_hostname              | 00401ba8-4fb0-4f1e-a7dc-e93065ebdd15            |
| hypervisor_type                  | ironic                                          |
| hypervisor_version               | 1                                               |
| id                               | 151                                             |
| uid                              | c9f98cc9-25e9-424e-8a89-002989054ec2            |
| updated_at                       |                                                 |
| vcpus                            | 48                                              |
| version                          | 78dbf26565cf24050718674dcf322331fab8ead5        |
+----------------------------------+-------------------------------------------------+
```

Any of the property listed in the field column may be used to reserve the nodes. For example, you can use `resource_properties='["=", "$architecture.smp_size", "2"]'` to reserve a node with two physical processors.

> **Note**
>
> Remember to use `$` in front of the property.

15.2.3 Extending a lease

To extend your lease, use `lease-update` command, and provide time duration via `--prolong-for` switch. The format of the duration is a number followed by a letter specifying the time unit. `w` is for weeks, `d` is for days and `h` is for hours. For example, if you would like to extend the `my-first-lease` by one day, run the following command:

```
openstack reservation lease update --prolong-for "1d" my-first-lease
```

15.2.4 Chameleon node types

The following node types are reservable on Chameleon.

Node Type	`resource_properties='["=", "$node_type", "<Chameleon node type name>"]'`
Skylake compute nodes	`compute_skylake`
Storage nodes	`storage`
Haswell Infiniband nodes	`compute_haswell_ib`
Storage Hierarchy nodes	`storage_hierarchy`
NVIDIA K80 nodes	`gpu_k80`
NVIDIA M40 nodes	`gpu_m40`
NVIDIA P100 nodes	`gpu_p100`
NVIDIA P100 NVLink nodes	`gpu_p100_nvlink`
NVIDIA RTX 6000 nodes	`gpu_rtx_6000`
FPGA nodes	`fpga`
Low power Xeon nodes	`lowpower_xeon`
Atom nodes	`atom`
ARM64 nodes	`arm64`

15.2.5 Creating a lease to reserve a VLAN segment

To create a lease, use the `lease-create` command. The following arguments are required:

- `--reservation` with the `resource_type` and `network_name` attributes
- `--start-date` in `"YYYY-MM-DD HH:MM"` format
- `--end-date` in `"YYYY-MM-DD HH:MM"` format
- A lease name

Optional attributes include `network_description` and `resource_properties` which can both be added to the `--reservation` argument.

For example, the following command will create a lease with the name of `my-first-vlan-lease` and the network name `my-network` that starts on June 17th, 2022 at 4:00pm and ends on June 17th, 2022 at 6:00pm:

```
openstack reservation lease create --reservation resource_type=network,network_name="my-
↪network" --start-date "2022-06-17 16:00" --end-date "2022-06-17 18:00" my-first-vlan-
↪lease
```

Adding the `network_description` attribute provides its value as the description field when creating the Neutron network for advanced networking configurations.

```
openstack reservation lease create --reservation resource_type=network,network_name="my-
↪network",network_description="OFController=${OF_CONTROLLER_IP}:${OF_CONTROLLER_PORT}" -
↪-start-date "2022-06-17 16:00" --end-date "2022-06-17 18:00" my-first-vlan-lease
```

Adding the `resource_properties` attribute allows you to reserve a specific *network segment* or *physical network* type. There are currently only two physical network types `physnet1` and `exogeni`. You can read more about both types in *Networking*. The following two examples show how to reserve a network by `segment_id` or `physical_network`.

```
openstack reservation lease create --reservation resource_type=network,network_name=my-
↪network,resource_properties='["=","$segment_id","3501"]' --start-date "2022-06-17 16:00
↪" --end-date "2022-06-17 18:00" my-first-vlan-lease
```

```
openstack reservation lease create --reservation resource_type=network,network_name=my-
↪network,resource_properties='["=","$physical_network","physnet1"]' --start-date "2022-
↪06-17 16:00" --end-date "2022-06-17 18:00" my-first-vlan-lease
```

While separate leases can be created to reserve nodes and VLAN segments, it is also possible to combine multiple reservations within a single lease. The following example creates a lease reserving one Skylake compute node and one VLAN segment:

```
openstack reservation lease create --reservation min=1,max=1,resource_type=physical:host,
↪resource_properties='["=", "$node_type", "compute_skylake"]' --reservation resource_
↪type=network,network_name="my-network" --start-date "2022-06-17 16:00" --end-date
↪"2022-06-17 18:00" my-combined-lease
```

Once your lease is active, the network named above already exists — you still need to configure a subnet and router on it before instances can use it. See *Isolated network VLANs* for those steps.

15.2.6 Creating a lease to reserve floating IPs

To create a lease, use the `lease-create` command. The following arguments are required:

- `--reservation` with the `resource_type` and `network_id` attributes
- `--start-date` in `"YYYY-MM-DD HH:MM"` format
- `--end-date` in `"YYYY-MM-DD HH:MM"` format
- A lease name

Multiple floating IPs can be reserved using the `amount` attribute. If ommitted, only one floating IP is reserved.

For example, the following command will create a lease with the name of `my-first-fip-lease` that starts on June 17th, 2022 at 4:00pm and ends on June 17th, 2022 at 6:00pm and reserves three floating IPs:

```
pip install python-openstackclient
PUBLIC_NETWORK_ID=$(openstack network show public -c id -f value)
openstack reservation lease create --reservation resource_type=virtual:floatingip,
↪network_id=${PUBLIC_NETWORK_ID},amount=3 --start-date "2022-06-17 16:00" --end-date
↪"2022-06-17 18:00" my-first-fip-lease
```

Once your floating IP lease is active, you can attach one of the reserved IPs to a running instance:

```
openstack server add floating ip <server> <floating-ip-address>
```

Use `openstack floating ip list` to see the floating IPs reserved by your lease, and `openstack server remove floating ip <server> <floating-ip-address>` to detach one later.

15.2.7 Reallocating a node in your lease

After creating your lease, you can view its details in the Horizon web interface. On this page, at the bottom, you can see a list of nodes in your lease. If you wish to reallocate one of the nodes in your lease, you can press the red "Re-Allocate Host" button next to it.

You can also do the same on the command-line. Run the command that follows, entering your lease ID and the node ID where appropriate.

```
openstack reservation host reallocate --lease-id LEASE_ID NODE_ID
```

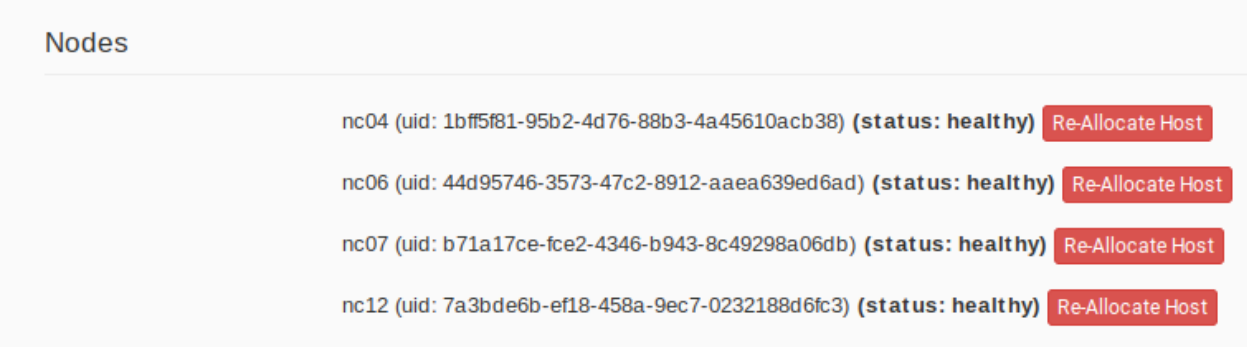


Fig. 12: The nodes on the lease detail page.

If you re-allocate a host because it is malfunctioning, please make sure to report it to the Help Desk so that we can fix it.

15.2.8 Creating a lease for a flavor (on KVM@TACC)

Since KVM@TACC is virtualized, instead of creating a lease for a physical host, you create a lease for a VM flavor. First, this requires finding what flavor you wish to create an instance of. You can find the list of available flavors by running the command:

```
openstack flavor list
```

which should return something like:

```
+--------------------------------------+---------------+--------+------+-----------+-----
↪--+-----------+
| ID                                   | Name          |    RAM | Disk | Ephemeral |␣
↪VCPUs | Is Public |
+--------------------------------------+---------------+--------+------+-----------+-----
↪--+-----------+
| 1                                    | m1.tiny       |    512 |    1 |         0 |    ␣
↪1 | True      |
| 2                                    | m1.small      |   2048 |   20 |         0 |    ␣
↪1 | True      |
| 3                                    | m1.medium     |   4096 |   40 |         0 |    ␣
↪2 | True      |
| 4                                    | m1.large      |   8192 |   40 |         0 |    ␣
↪4 | True      |
| 5                                    | m1.xlarge     |  16384 |   40 |         0 |    ␣
↪8 | True      |
| 6                                    | m1.xxlarge    |  32768 |   40 |         0 |   ␣
↪16 | True      |
| e81f12f8-de06-4107-acfd-e649217036ef | g1.h100.pci.1 | 250000 |   40 |         0 |   ␣
↪48 | True      |
+--------------------------------------+---------------+--------+------+-----------+-----
↪--+-----------+
```

Note the ID of the flavor you wish to create a lease for. Then to create a lease, use the `lease-create` command. The following arguments are required:

- `--reservation` with `resource_type=flavor:instance`, `flavor_id` and `amount` attributes

- `--start-date` in `"YYYY-MM-DD HH:MM"` format
- `--end-date` in `"YYYY-MM-DD HH:MM"` format

For example, the following command will create a lease with the name of `my-first-lease` for 2 `m1.large` instances that starts on June 17th, 2022 at 4:00pm and ends on June 17th, 2022 at 6:00pm:

```
openstack reservation lease create \
  --reservation "resource_type=flavor:instance,flavor_id=4,amount=2" \
  --start-date "2022-06-17 16:00" \
  --end-date "2022-06-17 18:00" \
  my-first-lease
```

Note

KVM@TACC lease durations follow a different policy than bare metal — see the Chameleon FAQ for details.

Once the lease is active, a new flavor named `reservation:<reservation-id>` becomes available — this is the flavor tied to your reservation. Launch your instance against it instead of the original flavor:

```
openstack server create \
  --flavor reservation:<reservation-id> \
  --image <image> \
  --network sharednet1 \
  my-kvm-instance
```

The `<reservation-id>` is the `id` of the individual reservation inside your lease (not the lease ID itself), which you can find with:

```
openstack reservation lease show <lease-name>
```

See *Work with KVM using the CLI* for other KVM@TACC CLI operations, such as managing security groups and creating instance snapshots.

CHAPTER

SIXTEEN

BARE METAL INSTANCES

Bare metal instances on Chameleon provide you with exclusive access to physical hardware resources. This allows you to run experiments with full control over the compute environment, from the hypervisor level down to the bare metal. Bare metal instances are ideal for systems research, performance evaluation, and experiments that require specific hardware features or configurations.

Before launching an instance, make sure you own a lease. About how to create a lease, see *Reservations*. Once your lease is started, you are almost ready to start an instance. But first, you need to make sure that you will be able to connect to it by setting up *Key pairs*.

Important

If your experiment doesn't need exclusive access to physical hardware, Chameleon also offers a multi-tenant, virtualized cloud via *KVM* and container-based edge computing via CHI@Edge (docs). For help deciding which fits your experiment, see the Tips and Tricks post Bare Metal or KVM? Which Should You Choose and When.

Note

Instances can also be launched and managed programmatically via the chi.server module in python-chi — see our *Jupyter and python-chi guide* for an introduction.

16.1 Launching instances with the GUI

16.1.1 Launch an instance

To launch an instance with the GUI, follow the steps:

1. In the navigation side bar, click *Project* > *Compute* > *Instances* to get to the *Instances* page.
2. Click the *Launch Instance* button in the upper right corner. This will open the *Launch Instance* wizard with several configuration steps. Steps with * are required.
3. In the *Details* step, enter a name for your instance that is unique within your project and select a currently active reservation for the instance.
4. In the *Source* step, select an image for your instance and click the "up" arrow. The image should move to the *Allocated* list, and can be removed by clicking the "Down" arrow if you wish to select a different image.
5. In the *Networks* step, select a network by clicking the "up" arrow next to it. To learn about the Chameleon default network and how to create your own network, see *Networking*.

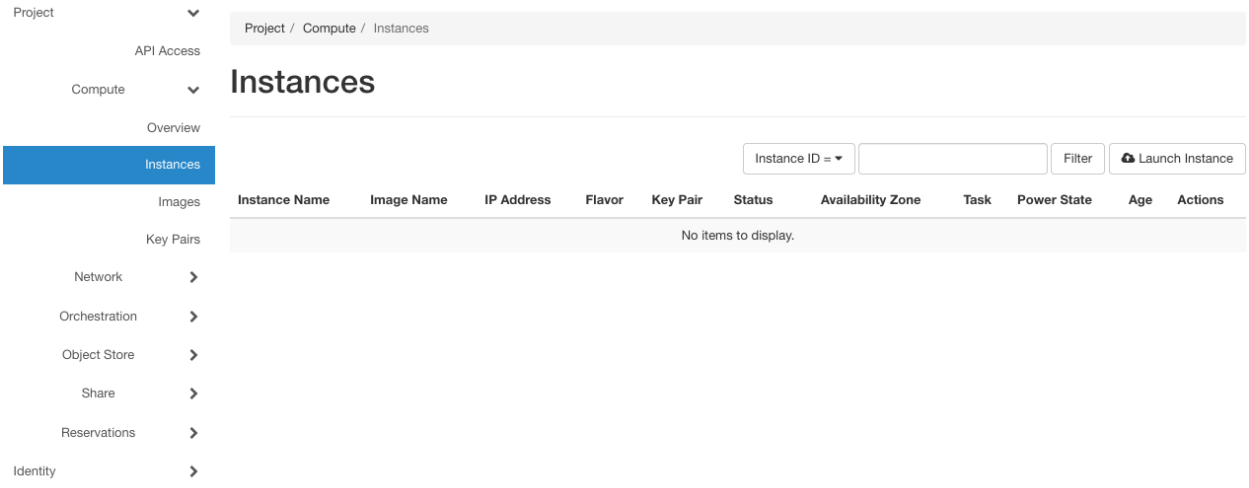


Fig. 1: The Instances page

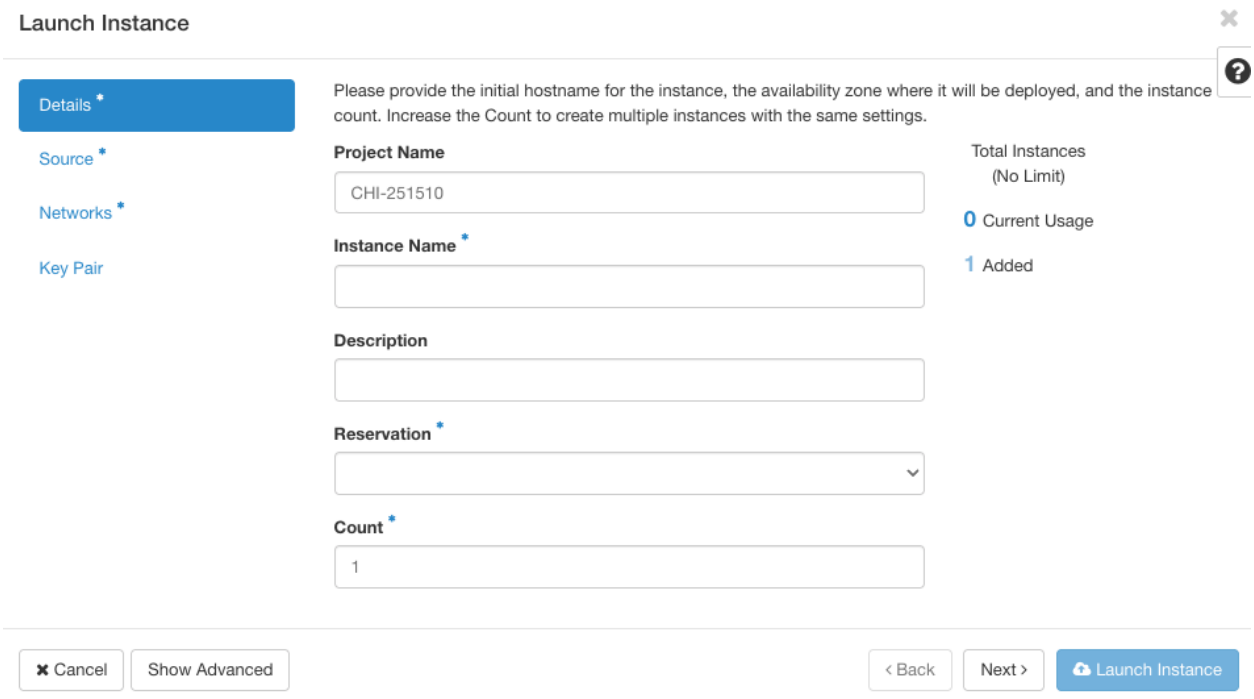


Fig. 2: The Launch Instance wizard.

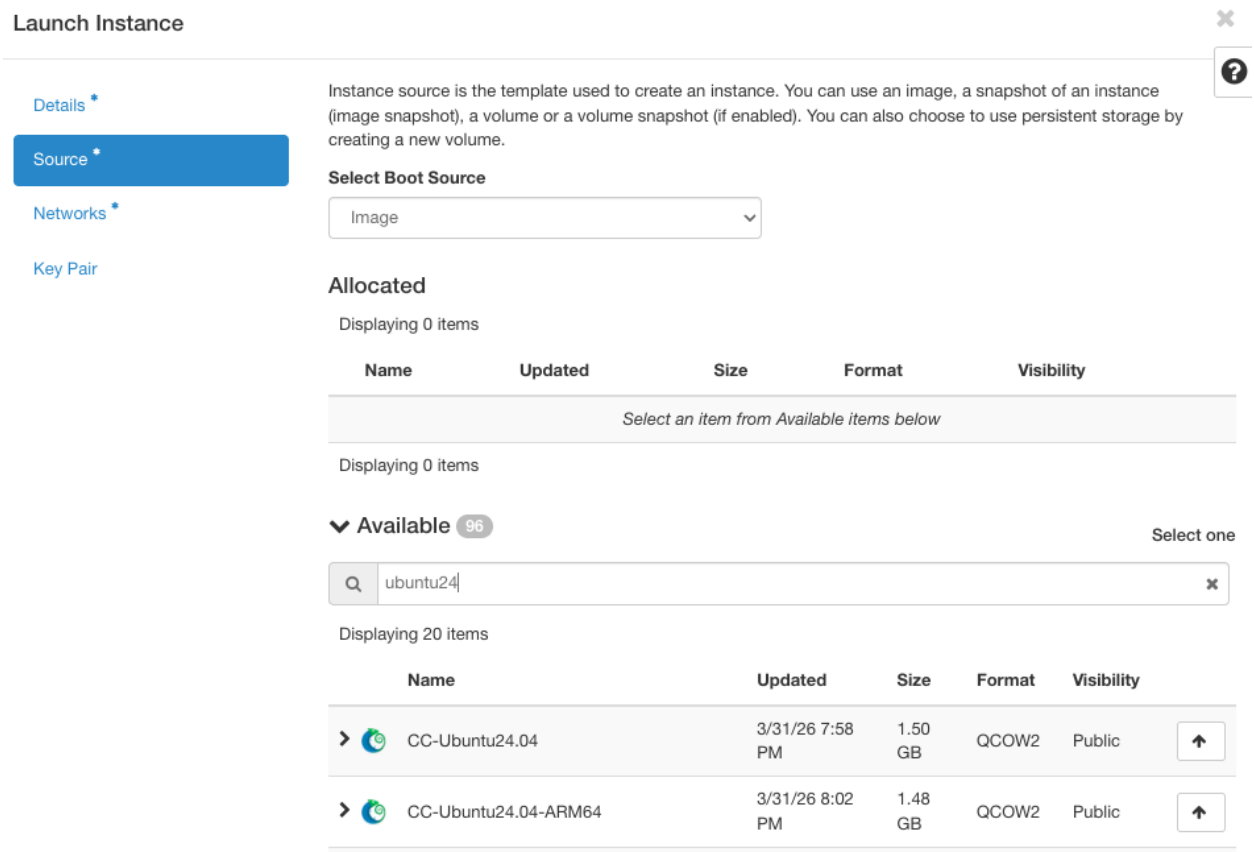


Fig. 3: The Source configuration step

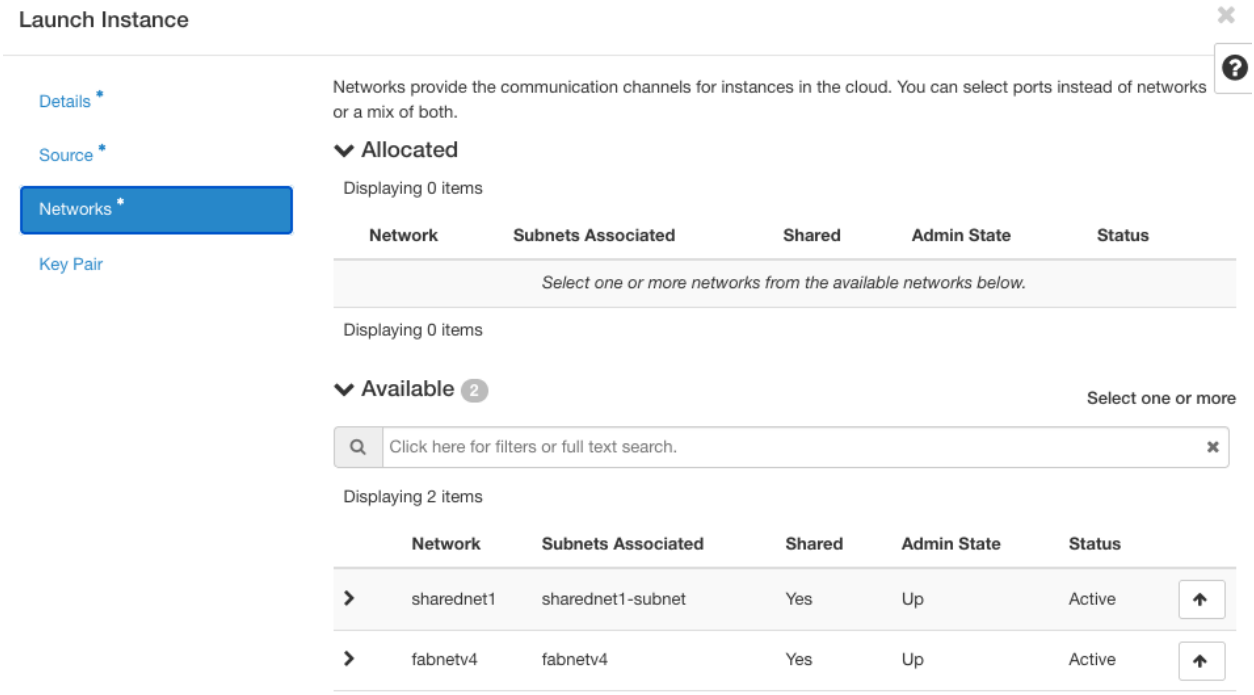


Fig. 4: The Source configuration step

6. In the *Key Pair* step, select one of your SSH key pairs. If you only have one key pair associated with your account, then it is selected by default.

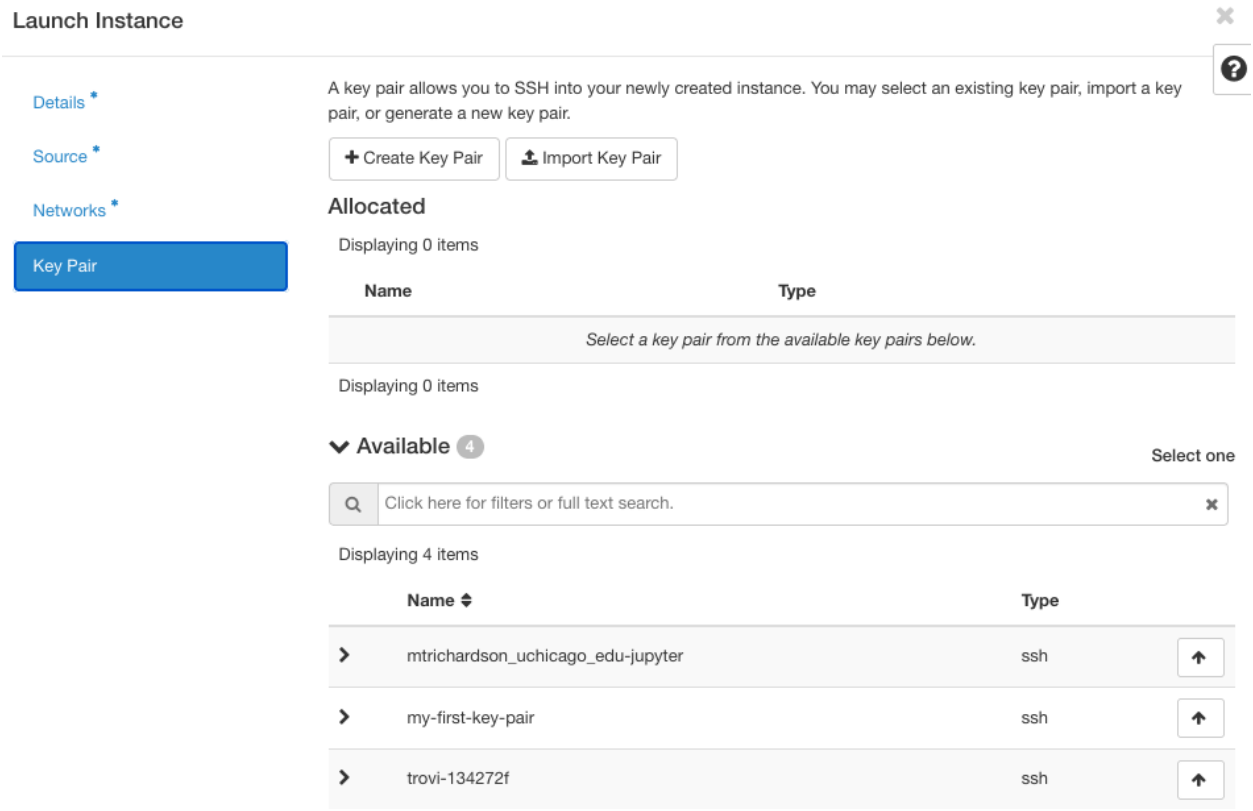


Fig. 5: The Key Pair configuration step

Important

It is a good practice to make sure that the instance is launching with the key pair of your choice, or you will not be able to access your instance.

Tip

You may import or create key pairs directly through this step.

Creating a New Key Pair

To create a key pair through the interface:

1. Click + *Create Key Pair* button
2. Provide a name for your new key pair and click *Create Key Pair*
3. A `.pem` file containing the private key will be automatically downloaded
4. The public key is saved automatically to Chameleon
5. Save the private key to a secure location (your home directory is recommended for macOS/Linux)

Importing an Existing Key Pair

To import a key pair you've generated on your computer:

1. Click *Import Key Pair* button
2. Provide a name for your imported key pair
3. Paste your public key (typically found at `~/.ssh/id_rsa.pub`)

> **Note**
>
> Chameleon **only** stores the public key for each SSH key pair. **Never** upload your private key! Private keys begin with `-----BEGIN RSA PRIVATE KEY-----`

> **Tip**
>
> On macOS, you can copy your public key with: `cat ~/.ssh/id_rsa.pub | pbcopy`

7. If you want to customize your instance after it has launched, you click *Advanced* and add a customization script in the *Configuration* step.
   - You can type in the script in *Customization Script*.
   - Or you can upload your script via *Load script from a file*.

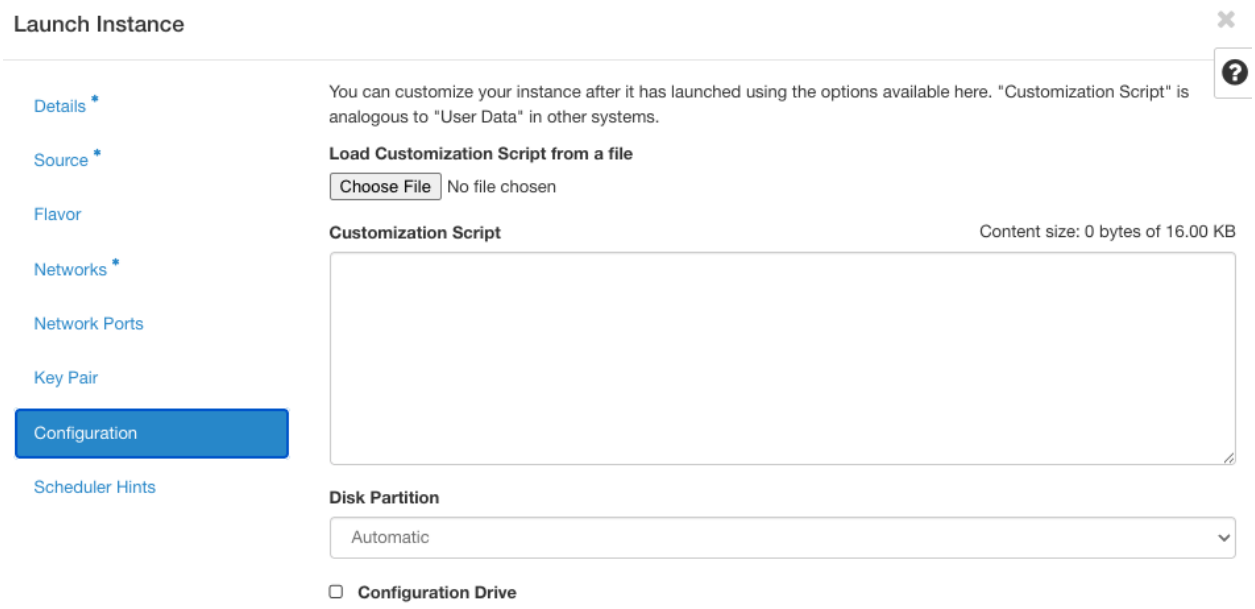


Fig. 6: Adding a Customization Script

8. Finish configuring and start launching the instance by clicking on the *Launch Instance* button. The instance will show up in the instance list, at first in *Build* status. It takes a few minutes to deploy the instance on bare metal hardware and reboot the machine.
9. After a few minutes, the instance should become *Active*. The power state will show as *Running*. You can now *Associate a floating IP*.

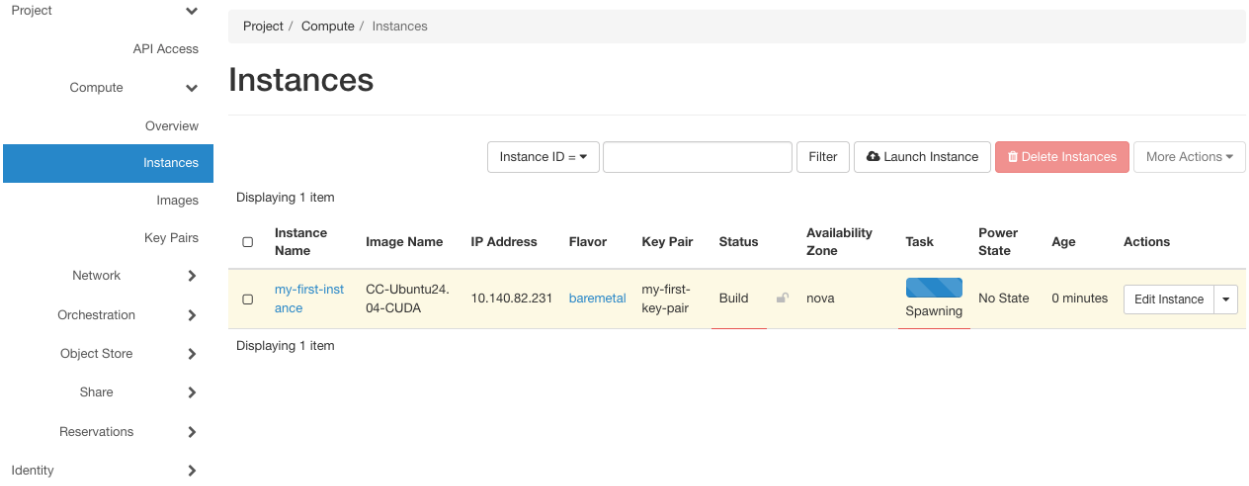


Fig. 7: An Instance with the Build status

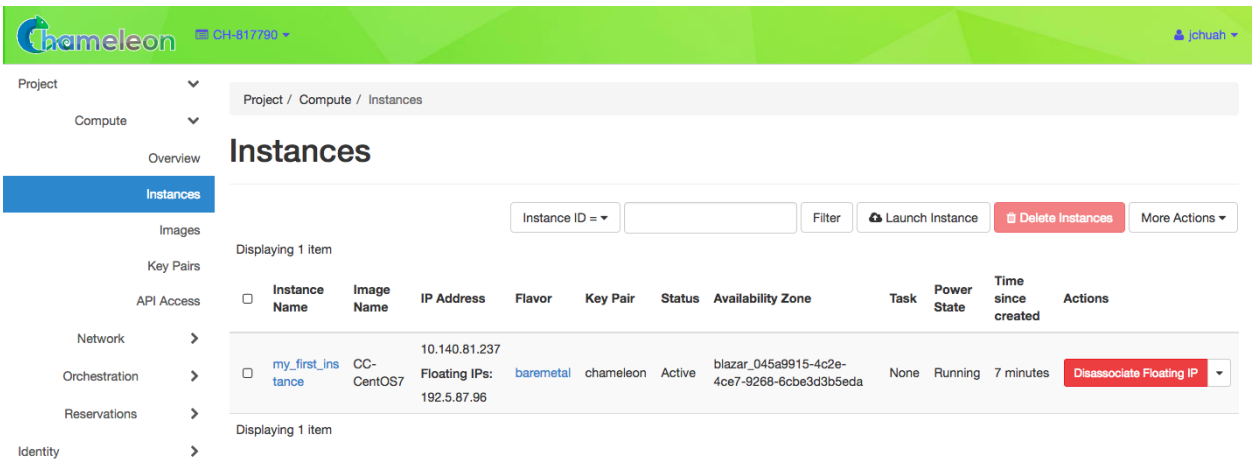


Fig. 8: An Instance with the Active status

10. To view instance details, click on the instance.

16.1.2 Associate a floating IP

To make your instance publicly accessible over the Internet, you must associate a *Floating IP Address* to it.

1. On the *Floating IPs* page (under the *Network* section in the left-hand sidebar), ensure that there is a free Floating IP available in your project. If there is not, click the *Allocate IP to Project* button to bring up the *Allocate Floating IP* dialog. In this dialog, you may simply click *Allocate IP*. You can optionally specify a description for the IP for your convenience.
2. Once a Floating IP is allocated to your project, it will display in the list view, and you can click the *Associate* button for the Floating IP to assign it to a running or spawning instance. This button will bring up the *Manage Floating IP Associations* dialog.
3. In the dialog, select an instance from the "Port to be associated" dropdown. Your instance's display name will be displayed here. Click *Associate* to complete the process of assigning the IP to your instance.
4. If you go back to the *Instances* page, you should now see the *floating IP* attached to the instance.

16.2 Launching instances with the CLI

Tip

Reading *Command Line Interface (CLI)* is highly recommended before continuing on the following sections.

16.2.1 Creating an instance with the CLI

To launch an instance inside a reservation, run:

```
openstack server create \
--image CC-CentOS8 \
--flavor baremetal \
--key-name <key_name> \
--nic net-id=<sharednet1_id> \
--hint reservation=<reservation_id> \
my-instance
```

The ID of the `sharednet1` network can be obtained using the command:

```
openstack network list
```

Alternatively, you may look it up in the GUI in the *Network* > *Networks* page. You can obtain your *reservation ID* via the GUI or by running:

```
openstack reservation lease show <lease_name>
```

Attention

The **reservation ID** and the **lease ID** are different

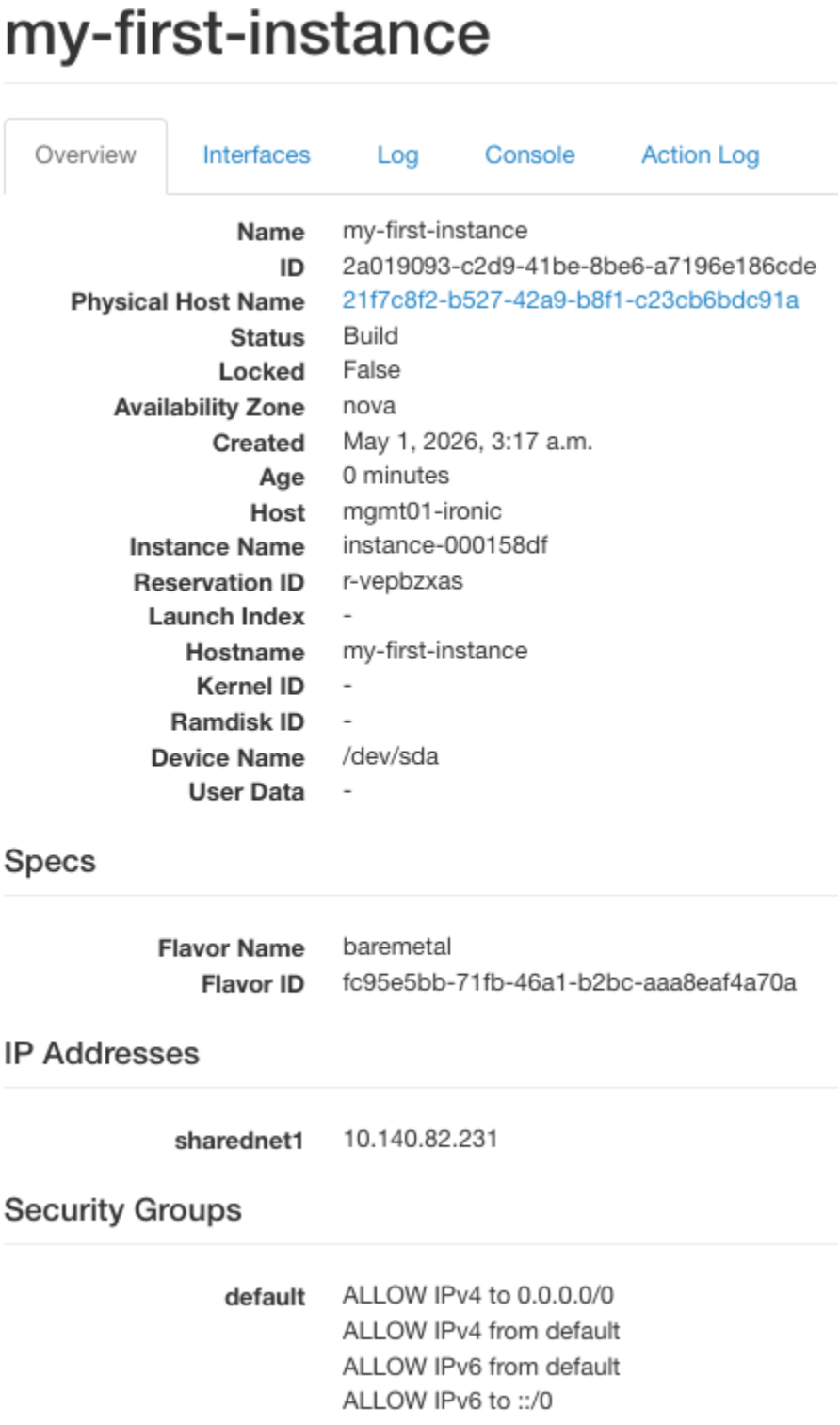


Fig. 9: Instance details

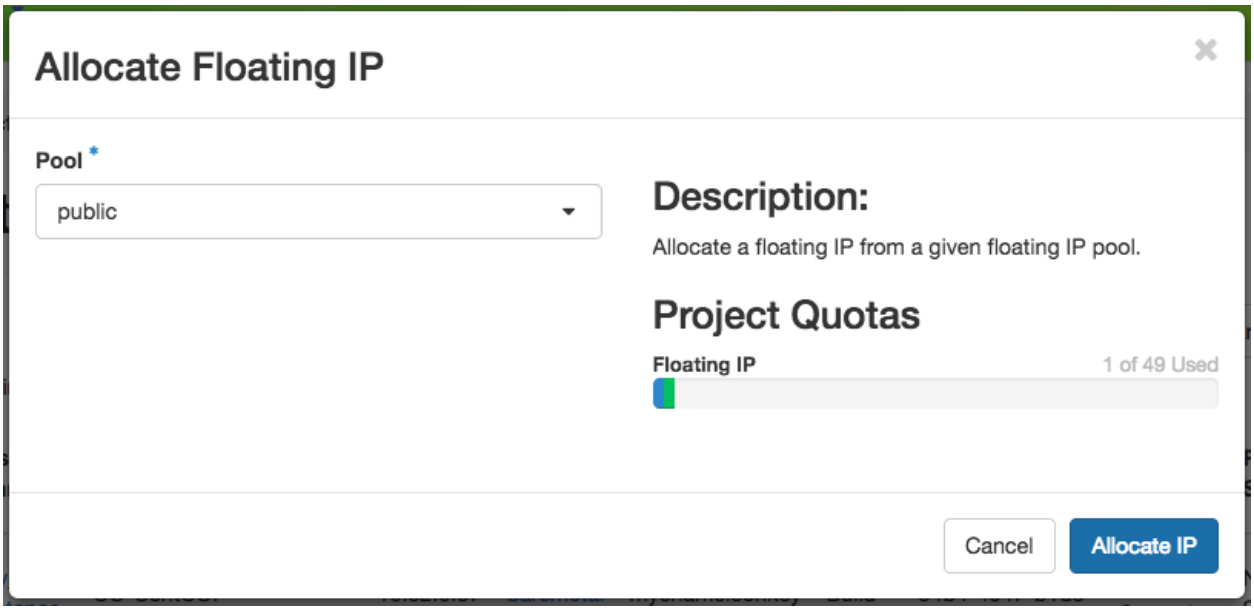


Fig. 10: The Allocate Floating IP dialog

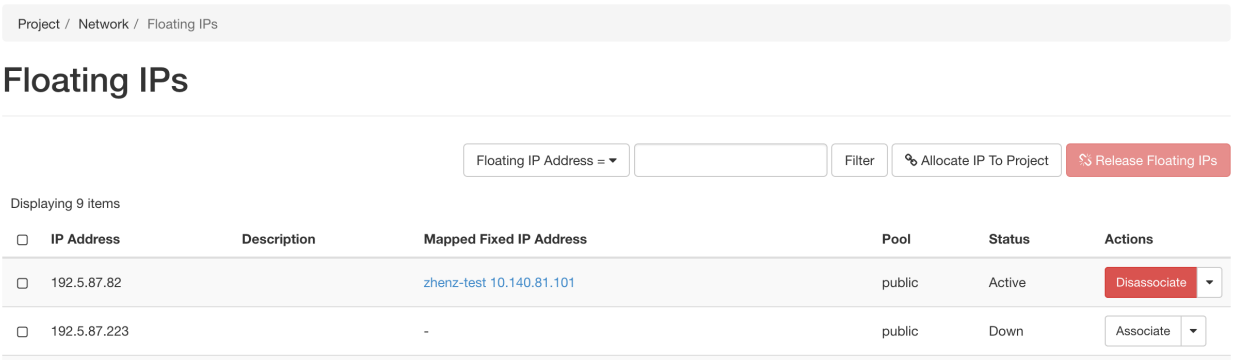


Fig. 11: The Floating IP list view with a Floating IP available

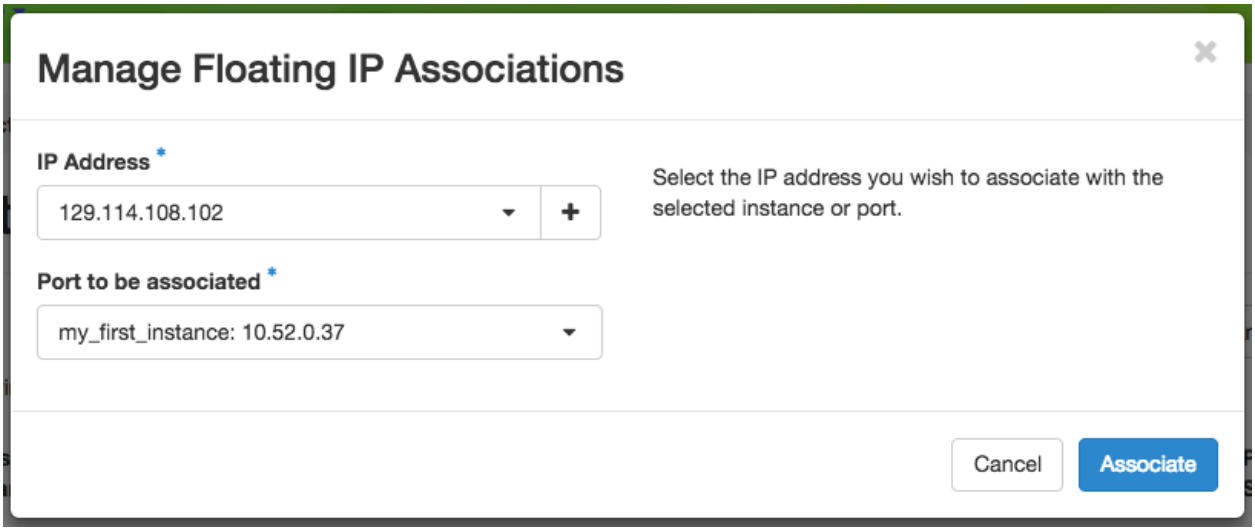


Fig. 12: The Manage Floating IP Associations dialog with an IP selected

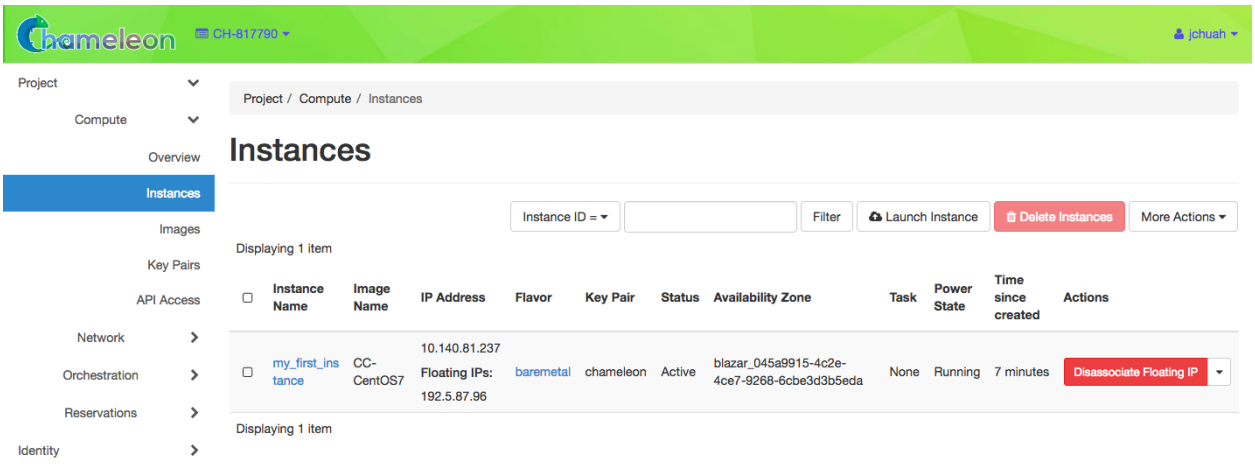


Fig. 13: An instance with an allocated Floating IP

Running a script on boot

You might want to automatically execute some code after launching an instance, whether it is installing packages, changing configuration files, or running an application. OpenStack provides a mechanism called User Data to pass information to instances. This information can be any data in any format, but if it is a shell script it will be automatically executed after boot by cloudinit. You can provide this shell script either via the GUI in the *Configuration* tab when launching an instance, or by providing a file to the nova command line using the `--user-data` option.

16.2.2 Customizing the kernel

It is easy to customize the operating system kernel or modify the kernel command line. You now have the option of modifying the boot loader configuration (e.g., `/boot/grub2/grub.cfg` on CentOS 7 images) to point it to a new kernel on the local disk, or specifying kernel parameters and then rebooting using this modified configuration.

To do this, you must be using a whole disk image rather than a partition image. Whole disk images contain their own kernel and ramdisk files and do not have `kernel_id` and `ramdisk_id` properties in the image repository, unlike partition images. Most Chameleon base images are whole disk images, giving you a good place to start if you're interested in custom kernels.

16.2.3 Running virtual machines on bare metal

For cloud computing and virtualization experiments, you might want to run virtual machines on bare hardware that you fully control rather than use the shared OpenStack KVM cloud. There are many different ways to configure networking for virtual machines. The configuration described below will enable you to connect your virtual machines to the Internet using a KVM Public Bridge which you must first configure manually on your host on the default network interface.

First, set up your environment for the OpenStack command line tools by following *the instructions*. Install the Neutron client in a virtualenv with:

```
pip install python-neutronclient
```

Then, for each virtual machine you want to run, request a Neutron port with:

```
openstack port-create sharednet1
```

This should display, among other information:

- A fixed IP in the same private network as the physical nodes
- A MAC address

Finally, start your virtual machine while assigning it the *MAC address* provided by OpenStack. If your image is configured to use *DHCP*, the virtual machine should receive the allocated IP.

Neutron ports allocated this way are not automatically deleted, so delete them after your experiment is over using:

```
openstack port delete <id>
```

You may find the ID of your ports using:

```
openstack port list
```

16.2.4 Inspecting your node

From within an instance you have already launched, you can discover which node it is running on by executing

```
curl http://169.254.169.254/openstack/latest/vendor_data.json
```

This will return a JSON dictionary describing site, cluster, and node.

16.2.5 Customizing networking

In its default configuration, the bare metal deployment system used by Chameleon (OpenStack Ironic <https://docs.openstack.org/ironic/latest/>`_) is restricted to using a single shared network per site. The network configuration features available in the dashboard are not supported (Networks and Routers). On CHI@UC, network layer 2 isolation is optionally available for compute nodes. You may find more details on the documentation for *Networking*.

16.3 Interacting with instances

Once your bare metal instance has launched, you may interact with it by using SSH if you have associated a *Floating IP* to it or by using the *Serial Console* from the GUI.

Note

You can also connect to and run commands on instances programmatically via the chi.ssh module in python-chi — see our *Jupyter and python-chi guide* for an introduction.

16.3.1 Connecting via SSH

If you have associated a *Floating IP* with the instance and you have the private key in place, you should be able to connect to the instance via SSH using the `cc` account.

To access the instance using SSH, type the command in your terminal:

```
ssh cc@<floating_ip>
```

Error

If you get errors:

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
...
```

It is likely that you have saved a previous entry for the instance's *Floating IP* in your `~/.ssh/known_hosts` file on your computer. Simply removing the entry from the file should solve the issue.

You can remove the entry from the `~/.ssh/known_hosts` file by using the command:

```
ssh-keygen -R <floating_ip>
```

You may receive the response below. Type `yes` and hit enter:

```
The authenticity of host '130.202.88.241 (130.202.88.241)' can't be␣
↪established.
RSA key fingerprint is 5b:ca:f0:63:6f:22:c6:96:9f:c0:4a:d8:5e:dd:fd:eb.
Are you sure you want to continue connecting (yes/no)?
```

When logged in, your prompt may appear like this:

```
[cc@my-first-instance ~]$
```

Note

If you notice SSH errors such as connection refused, password requests, or failures to accept your key, it is likely that the physical node is still going through the boot process. In that case, wait before retrying. Also make sure that you use the `cc` account. If after 10 minutes you still cannot connect to the machine, open a ticket with our Help Desk.

You can now check whether the resource matches its known description in the resource registry. For this, simply run:

```
sudo cc-checks -v
```

The `cc-checks` program prints the result of each check in green if it is successful and red if it failed. You can now run your experiment directly on the machine via SSH. You can run commands with root privileges by prefixing them with `sudo`. To completely switch user and become root, use the `sudo su - root` command.

Attention

`cc-checks` is only available on legacy CentOS7 images!

16.3.2 Connecting via serial console

Chameleon now allows you to connect to the serial console of your bare metal nodes via the GUI. Once your instance is deployed, click on the *Console* button in the instance contextual menu.

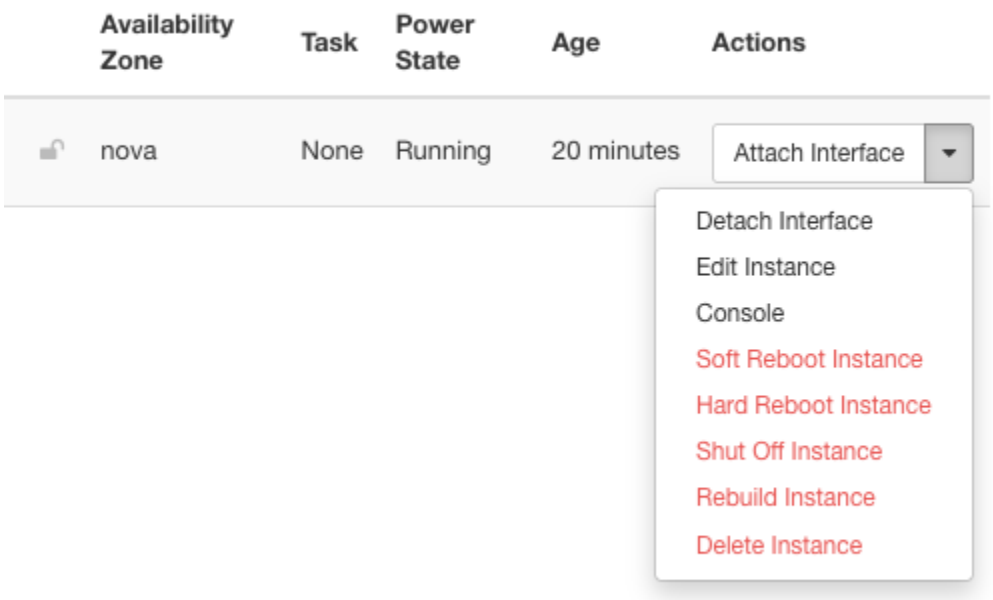


Fig. 14: The serial console button

This should open a screen showing an interactive serial console (it could take some time to show up, give it 30 seconds or so).

Our latest images are configured to auto-login into the `cc` account. Other images may show you a login prompt. You can set a password on the `cc` account by accessing it via SSH, using the command `sudo passwd cc`, and then using this password to connect to the console.

16.4 Composable Hardware

Composable hardware lets you change which GPUs are attached to which bare metal nodes *after* you reserve them, instead of being locked into whatever GPU-to-node pairing shipped in the chassis. Chameleon's composable systems are built on GigaIO's **FabreX** technology: a configurable PCIe backplane that acts as a virtual switch, so an accelerator attached to any node on the fabric appears to that node's OS as if it were directly, physically connected.

Because FabreX connects accelerators over PCIe rather than through a host's memory controller, composable nodes also support device-to-device communication, enabling Nvidia GPUDirect peer-to-peer transfers directly between GPUs.

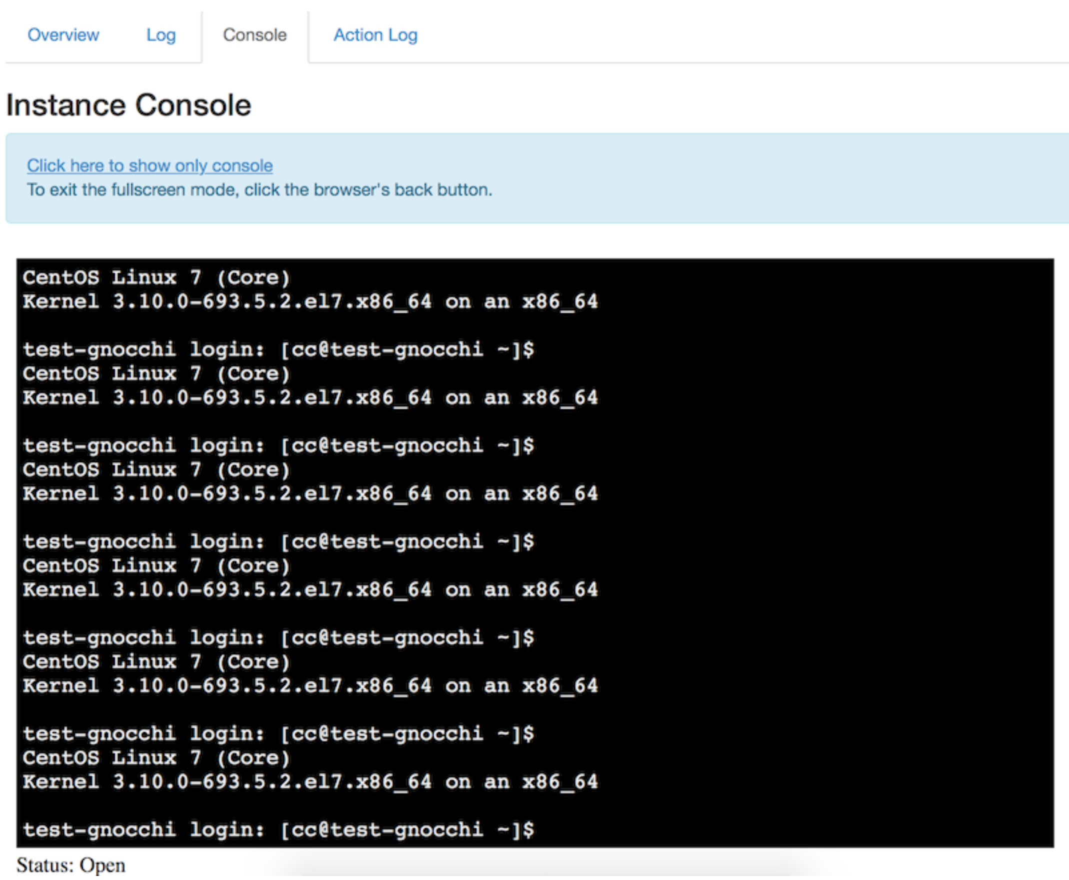


Fig. 15: An open console

> **Tip**
>
> For a walkthrough of the FabreX architecture and why composability is useful for GPU-hungry experiments, see the Tips and Tricks post Composable Hardware on Chameleon NOW!.

16.4.1 Hardware specifications

Composable GigaIO nodes are filterable in the *Resource Browser* and reservable by the `compute_gigaio` node type.

CHI@UC

- 8x Dell PowerEdge R6525 nodes
- 2x AMD EPYC 7763 processors per node (256 threads per node)
- 512 GB RAM per node
- 2x 480 GB SATA SSDs per node
- 8x Nvidia A100 PCIe GPUs (80 GB), one attached to each node by default

CHI@TACC

- 6x Dell PowerEdge R650 nodes
- Intel Xeon Platinum 8380 processor per node
- 256 GB RAM per node
- 4x Nvidia A100-class PCIe GPUs, distributed across the 6 nodes: 2x A100 (40 GB) and 2x A30

Note

CHI@TACC also hosts a separate LIQID-based composable system (`compute_liqid` node type: 8x PowerEdge R6525 nodes with 10x A100 PCIe GPUs and NVMe storage on the fabric). It uses the same reservation and recomposition workflow described below.

16.4.2 Reserving and recomposing nodes

Each composable node comes with one GPU attached by default, so to compose a system with multiple GPUs on a single node you must reserve all of the nodes whose GPUs you want to combine. For example, to end up with 4 GPUs on one node, reserve 4 `compute_gigaio` nodes.

1. Create a lease for the `compute_gigaio` node type as you would for any other bare metal resource — see *Reservations*. For example, via the CLI:

```
openstack reservation lease create \
  --reservation min=4,max=4,resource_type=physical:host,resource_properties='["=", "
↪$node_type", "compute_gigaio"]' \
  --start-date "2026-07-17 16:00" \
  --end-date "2026-07-18 16:00" \
  my-composable-lease
```

2. Once your lease is confirmed, open a ticket at the Help Desk requesting the GPU-to-node configuration you need. Chameleon staff will recompose the fabric to match your request when your reservation starts — recomposition is not self-service and does not happen automatically.

Attention

Submit your recomposition ticket as soon as your lease is confirmed. Recomposition happens when your reservation *starts*, so a ticket filed after that point may delay access to the hardware for the beginning of your reservation window.

CHAPTER

SEVENTEEN

IMAGES

All instances in Chameleon, whether KVM or bare metal, are running off disk images. The content of these disk images can be snapshotted at any point in time, which allows you to save your work and launch new instances from updated images later. While OpenStack KVM has built-in support for snapshotting in the Horizon web interface and via the command line, bare metal instances require a more complex process.

To work around this limitation, we provide the `cc-snapshot` utility that you can execute from inside your running instance. The `cc-snapshot` utility is pre-installed in all Chameleon supported appliances. You can find our appliances on Trovi by filtering for the **appliance** tag.

Note

Images can also be managed programmatically via the chi.image module in python-chi — see our *Jupyter and python-chi guide* for an introduction.

The image service on Chameleon uses OpenStack Glance. This documentation demonstrates how to accomplish common tasks with *Images* using the GUI and the CLI.

The following matrix shows which Chameleon-supported images are compatible with the hardware available at each site. For example, CHI@UC does not have nodes with the ARM64 architecture, so the ARM64 image variants are not useful at that site.

Supported images by site:

Image	CHI@TACC	CHI@UC	KVM@TACC	CHI@NCAR
`CC-CentOS9-Stream`	Yes	Yes	Yes	Yes
`CC-Ubuntu22.04`	Yes	Yes	Yes	Yes
`CC-Ubuntu22.04-ARM64`	Yes	No	No	Yes
`CC-Ubuntu22.04-CUDA`	Yes	Yes	Yes	No
`CC-Ubuntu24.04`	Yes	Yes	Yes	Yes
`CC-Ubuntu24.04-ARM64`	Yes	No	No	Yes
`CC-Ubuntu24.04-CUDA`	Yes	Yes	Yes	No
`CC-Ubuntu24.04-CUDA-VGPU`	No	No	Yes	No
`CC-Ubuntu24.04-ROCm`	Yes	No	No	No
`CC-Ubuntu26.04`	Yes	Yes	Yes	Yes

17.1 Chameleon supported images

There are a number of images built and supported by the Chameleon team, specifically:

- CC-Ubuntu26.04
- CC-Ubuntu24.04
- CC-Ubuntu24.04-CUDA
- CC-Ubuntu24.04-CUDA-VGPU
- CC-Ubuntu24.04-ROCm
- CC-Ubuntu24.04-ARM64
- CC-Ubuntu22.04
- CC-Ubuntu22.04-CUDA
- CC-Ubuntu22.04-ARM64
- CC-Centos9-Stream

The CUDA images, such as Ubuntu24.04-CUDA, contain various settings, software, and drivers specifically for NVIDIA GPU nodes. The `CC-Ubuntu24.04-CUDA-VGPU` image is a variant of the CUDA image for virtual GPU (vGPU) nodes at KVM@TACC. It is intended for use with the `g1.h100.vgpu.1g.12gb` flavor and comes pre-configured with the appropriate vGPU license. The ROCm images contain similar settings, software, and drivers for AMD GPU nodes. And finally, the ARM64 images, such as Ubuntu24.04-ARM64, are images specifically built with ARM support for ARM nodes. Non-ARM images all assume x86-based architectures. All Chameleon-supported images can be found on Trovi under the **appliance** tag.

> **Warning**
>
> Any images with operating system versions that are end-of-life, such as Ubuntu18.04 are no longer offered as Chameleon-supported images. However, they can still be used on Chameleon with caution. Please note that these images are EOL so they no longer receive security updates and bug fixes. They may stop working at any point, therefore, make sure to upgrade and move your environments to newer versions of the operating system as soon as possible. If you do need to launch older images on Chameleon, please consider using a bastion host that is up-to-date and provides SSH access. From there, you can jump to your instances running older images that are not exposed directly on the Internet.

You may also build your own images and share them with the community. These images may be built from scratch or based on the Chameleon-supported images. However, the Chameleon team cannot offer support for user-provided images.

17.2 The `cc-snapshot` utility

17.2.1 Overview

The `cc-snapshot` utility implements snapshotting a bare metal instance from command line and uploads it to Glance, so that it can be immediately used to boot a new bare metal instance. The snapshot images created with this tool are whole disk images.

For ease of use, `cc-snapshot` has been installed in all the appliances supported by the Chameleon project. If you would like to use it in a different setting, it can be downloaded and installed from the github repository.

Tip

`cc-snapshot` is also a useful tool for preserving an instance before it expires. See the Tips and Tricks post Extending Your Research Artifacts' Lifespan for how default resource retention periods work and other ways to extend them.

Note

As of the *vendordata authentication changes*, `cc-snapshot` no longer reads credentials from vendordata. You must have OpenStack credentials in your environment first — for example by running `ccauth openrc --output ~/openrc && source ~/openrc` — and then run `cc-snapshot` with `sudo -E` so root inherits them.

To make a snapshot of a bare metal instance, run the following command from inside the instance:

```
sudo -E cc-snapshot <image_name>
```

Tip

You may get warnings, such as "image too large", during snapshotting, and get prompted to confirm. If you are confident about what you are trying to do, you can skip all warnings by using the `-f` flag.

```
sudo -E cc-snapshot -f <image_name>
```

In addition, you can exclude directories by using the `-e` flag.

```
sudo -E cc-snapshot -e <dir1> -e <dir2> <image_name>
```

You can use the faster zstd compression (default is zlib) by using the `-z` flag. Note that launching images compressed with zstd won't work on sites running OpenStack Xena or earlier. All core sites like CHI@UC and CHI@TACC are up-to-date, but some associate sites, such as CHI@NU, may fail to launch new instances with a zstd compressed image. If you are using an associate site, we suggest checking that you can launch a new instance with the image from your snapshot. You can check the compression type of an image by running *openstack image show <image_uuid>* and checking the "compression_type" property. Images set to *zstd* are using the new method, and images without a *compression_type* property or with it set to *zlib* are using the old method.

```
sudo -E cc-snapshot -z <image_name>
```

To see all available options for `cc-snapshot`, run `sudo -E cc-snapshot -h`.

Tip

See *Changes to automatic credential setup on instances* for how to generate credentials with *ccauth* or the *The OpenStack RC script* before running `cc-snapshot`.

Note

When using the `cc-snapshot`, it will create an image within your project with the `shared` visibility. Anyone with access to your project can access this image.

Note

If you choose an *Image* name that already exists, the previous one **will not** be overwritten. A new *Image* with the same name but a different *UUID* will be generated.

Note

If you install a custom kernel, make sure the size of your running kernel (`/lib/modules/<kernel_version>`) is less than 4GB. To find out which kernel version you're running, run `uname -r`.

17.2.2 Updating cc-snapshot

Error

If you receive the following error:

```
public endpoint for image service in regionOne not found Unable to contact Glance,␣
↪check username and password
```

it means that you have an outdated copy of `cc-snapshot` and you will need to update `cc-snapshot`. This usually happens when you use an older images that contains an outdated version of `cc-snapshot`.

You may also want to get new functionalities added to the latest version of `cc-snapshot`.

Run the following commands from your instance:

```
curl -O https://raw.githubusercontent.com/ChameleonCloud/cc-snapshot/master/cc-
↪snapshot
sudo mv cc-snapshot /usr/bin/
sudo chmod +x /usr/bin/cc-snapshot
```

17.3 Managing images using the GUI

To manage your images, use the *Images* page at CHI@TACC or CHI@UC, by clicking on *Project* > *Compute* > *Images*.

Note

The Chameleon logo next to an image's name indicates that this image is an appliance supported by the Chameleon project. Chameleon-supported appliances can be found on Trovi by filtering for the **appliance** tag.

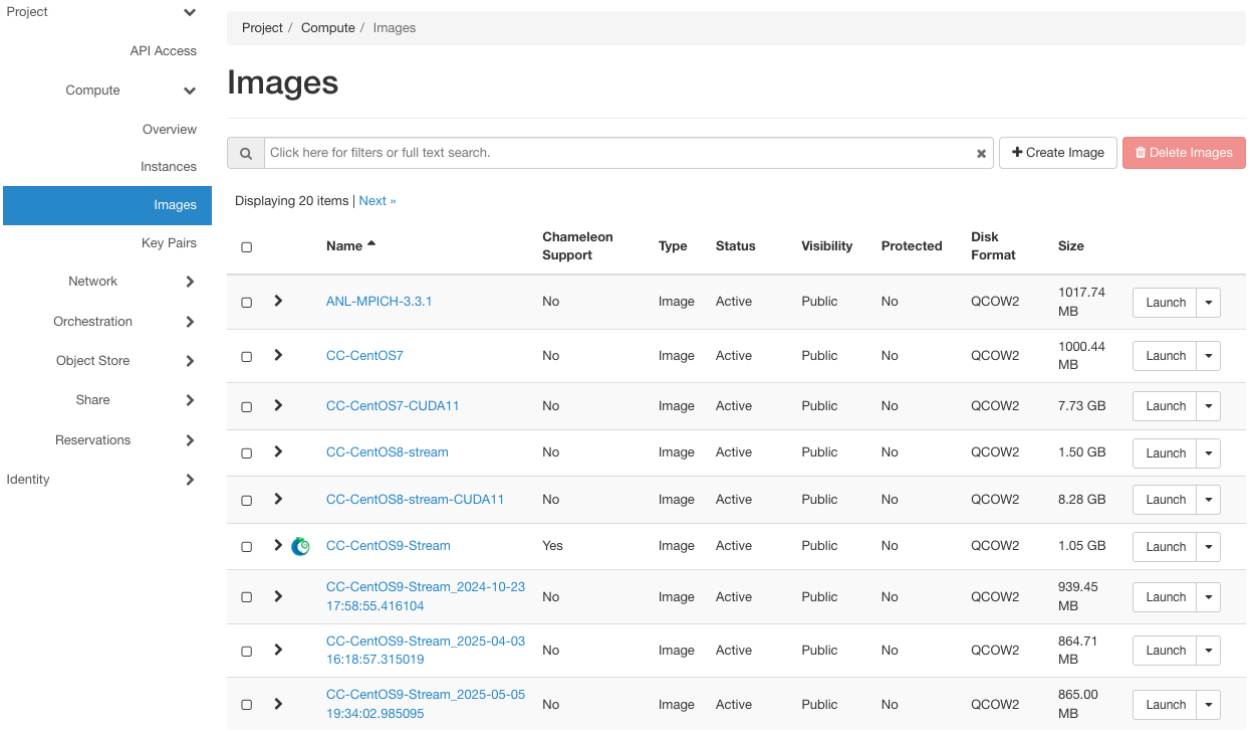


Fig. 1: The Images page

> **Note**
>
> Images at each site are stored independently. An Image made at CHI@TACC **will not** be available at CHI@UC (or vice versa) unless transferred manually.

17.3.1 Uploading an image

Use + *Create Image* button to upload an image.

In the *Create Image* dialog:

1. Enter an *Image Name* and, optionally, a description.
2. Click *Browse* to select a file on your local machine to upload.
3. Select a *Format* of the image. Images created by the `cc-snapshot` utility are *QCOW2* images.
4. To add additional metadata for your image, use the *Metadata* section by clicking *Metadata* in the sidebar.
5. Click the *Create Image* button to upload your image.

17.3.2 Launching instance using an image

During the process of *launching instance* from the *Instance* page, it will ask you to select an image. Alternatively, you can launch instances with a selected image from the *Image* page by simply clicking on the *Launch* button located in the same row of the targeted image.

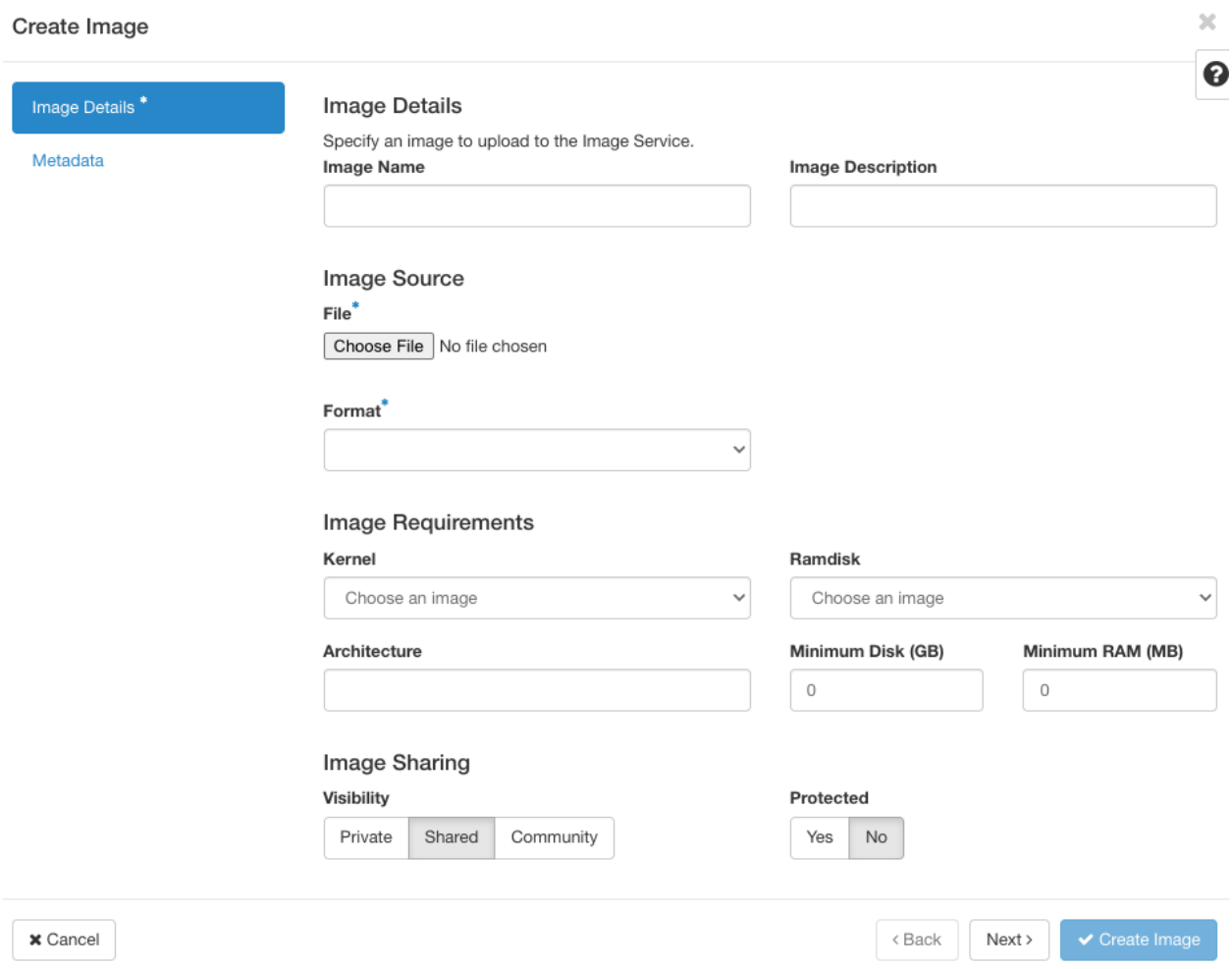


Fig. 2: The Create Image dialog

> **Tip**
>
> Other than *Launch*, there are other actions you may perform on the image. Clicking on the dropdown to explore more on what you can do.

17.3.3 Viewing image details

To view image details, click on the name of the Image.

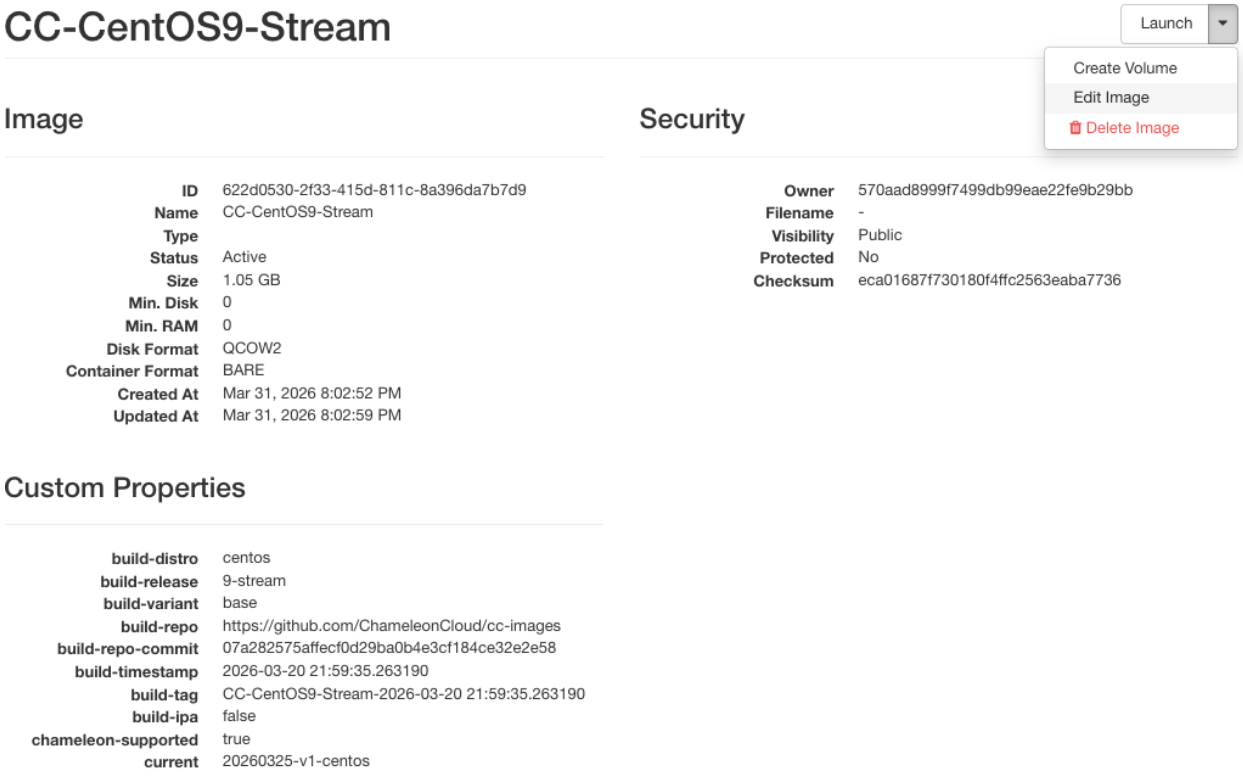


Fig. 3: Image details

The dropdown list in the top right corner allows you to perform various actions on the selected image, such as *Launch*, *Edit Image*, and *Update Metadata*.

> **Tip**
>
> The *ID* on the image details' page is useful when you work on the image using the CLI.

17.3.4 Publishing images as appliances via Trovi

Note

New appliances can no longer be submitted to the Appliance Catalog. Appliances are now published through Trovi.

To share an image as a Chameleon appliance, upload or locate your artifact in Trovi and open its **Edit** menu. Add the **appliance** tag along with a descriptive name, author and support contact information, version, and an informative description. Once published, your appliance will be discoverable by other Chameleon users when they filter Trovi by the **appliance** tag.

Entering a descriptive name and an informative description is encouraged. **The description is used by others to determine if an appliance contains the tools needed for their research.**

Tip

To make your description effective you may want to ask the following questions:

- What does the appliance contain?
- What are the specific packages and their versions?
- What is it useful for?
- Where can it be deployed and/or what restrictions/limitations does it have?
- How should users connect to it and what accounts are enabled?

17.4 Managing images using the CLI

Tip

Reading *Command Line Interface (CLI)* is highly recommended before continuing on the following sections.

17.4.1 Uploading an image

After configuring the environment variables using *The OpenStack RC script*, run the following command:

```
openstack image create --file <file> --disk-format <format> <image-name>
```

Provide the path to and the name of your image file in your local file system as the value of the `file` parameter. Also, indicate the image format using the `format` switch, such as `QCOW2`. Finally, name your image via the `image-name` switch.

17.4.2 Downloading an image

Downloading an image file to your local machine is **only** available via the CLI. You may find it useful when transferring images from one Chameleon site to another. To download an image file, run the following command:

```
openstack image save --file <filename> <image>
```

Use `filename` to indicate where you would like to save the image in your local file system. Also, replace `image` with either the name or the *ID* of the image on Chameleon.

Important

If you do not provide the `--file` parameter, it will print out the binary image data in your terminal.

17.4.3 Retrieving images

You may list all images of your project by typing:

```
openstack image list
```

Optionally, you may add filters to the list, such as `--shared` to only display the images shared within your project. Use `openstack image list --help` to see all the available filters.

17.4.4 Viewing image details

You may view details of an image with the command:

```
openstack image show <image>
```

Replace `image` with either an image name or it's *UUID*.

17.4.5 Sharing an image

You may share images several ways. If you wish to share an image with everyone, use:

```
openstack image set --public <image>
```

Replace `image` with the image *UUID*.

If you would like to share an image with another project, first set the image visibility to shared:

```
openstack image set --shared <image>
```

Next add the project you wish to share the image with:

```
openstack image add project <image> <project>
```

Replace `image` and `project` with the corresponding *UUIDs*

Finally the project that the image is shared to must accept the shared image. Run this command with a user in the second project:

```
openstack image set --accept <image>
```

Replace `image` with the image *UUID* and the second project should now be able to use the image!

Important

Only the owner of the image can modify it or any properties. However a project who has an image shared to it can remove themselves from the list of image members.

17.4.6 Editing an image

You may edit an image using the command:

```
openstack image set <image> ...
```

Replace `image` with either an image name or it's *UUID*. You must provide additional flags to update an image. Use `openstack image set --help` to see all the options.

CHAPTER

EIGHTEEN

POWER MONITORING

Chameleon provides comprehensive power monitoring capabilities to help researchers measure and understand the energy consumption of their experiments.

Tip

For detailed examples, tool installation instructions, and advanced techniques, see our Power Measurement and Management blog post.

18.1 Available power monitoring methods

Infrastructure-level monitoring: - Automatic power and temperature data collection via IPMI/DCMI - Works on most server-class Intel and AMD nodes - Provides system-level power consumption data

Application-level monitoring: - `etrace2`: Energy measurement for individual applications using Intel RAPL - `perf`: Quick RAPL energy measurements - Scaphandre: Advanced per-process power tracking

Long-term monitoring: - Prometheus exporters and Grafana for continuous data collection and visualization

18.2 Hardware support

Power monitoring support varies by node type: - **Full support**: Most Intel and AMD compute/GPU nodes - **Limited support**: Specialized nodes (FPGAs, ARM64) - **Temperature monitoring**: Only available when nodes are powered on

18.3 Getting started

1. **For system-level monitoring**: Use `ipmitool dcmi power reading` to get current power consumption
2. **For application-level monitoring**: Use `etrace2 <your_program>` to measure energy consumption of specific applications
3. **For detailed instructions**: See the power monitoring blog post

Note

Power monitoring tools use software-based estimation models and may include system overhead. For accurate measurements, consider baseline readings and validate with multiple tools when possible.

CHAPTER

NINETEEN

COMPLEX APPLIANCES

Deploying an MPI cluster, an OpenStack installation, or any other type of cluster in which nodes can take on multiple roles can be complex: you have to provision potentially hundreds of nodes, configure them to take on various roles, and make them share information that is generated or assigned only at deployment time, such as hostnames, IP addresses, or security keys. When you want to run a different experiment later you have to redo all this work. When you want to reproduce the experiment, or allow somebody else to reproduce it, you have to take very precise notes and pay great attention to their execution.

To help solve this problem and facilitate reproducibility and sharing, the Chameleon team configured a tool that allows you to deploy complex clusters with "one click". This tool requires not just a simple *image* (i.e., appliance) but also a document, called a *template*, that contains the information needed to orchestrate the deployment and configuration of such clusters. We call this *image* + *template* combination **Complex Appliances** because it consists of more than just the image (i.e., appliance).

In a nutshell, *Complex Appliances* allow you to specify not only what image you want to deploy but also on how many nodes you want to deploy that image, what roles the deployed instances should boot into (such as e.g., head node and worker node in a cluster), what information from a specific instance should be passed to another instance in that *Complex Appliance*, and what scripts should be executed on boot so that this information is properly used for configuring the "one click" cluster.

This guide will tell you all you need to know in order to use and configure *Complex Appliances* on Chameleon.

Hint

Since *Complex Appliances* in Chameleon are currently implemented using the OpenStack Heat orchestration service, we will be using OpenStack terminology and features to work with them. The templates described above are YAML files using the Heat Orchestration Template (HOT) format (Heat also supports the AWS CloudFormation template format, but this is not covered here). A deployed complex appliance is referred to as a "stack" – just as a deployed single appliance is typically referred to as an "instance".

Note

Multi-node deployments can also be orchestrated programmatically with python-chi instead of Heat templates — see *Advanced topics* for an example that combines python-chi with Ansible. For a worked example of building an MPI cluster this way, see the Tips and Tricks post Building MPI Clusters on Chameleon: A Practical Guide.

19.1 Complex appliances in Trovi

Note

All Chameleon appliances, including *Complex Appliances*, are now stored and shared exclusively through Trovi. Filter Trovi artifacts by the **appliance** tag to find Chameleon-supported OS images and heat templates. The old Appliance Catalog is deprecated and no longer used.

Trovi has several *Complex Appliances* for popular technologies that people want to deploy such as OpenStack or MPI or even more advanced deployments such as efficient SR-IOV enabled MPI in KVM virtual machines. We also provide common building blocks for cluster architectures, such as an NFS share.

To find *Complex Appliances*, go to Trovi and filter by the **appliance** tag. Click on any artifact to view its details, including the heat template needed to launch it.

Tip

You may download the *Template* file or copy the *Template* file URL to clipboard from the artifact's detail page on Trovi. The *Template* file or its URL is required when launching a *Complex Appliance*.

19.2 Managing complex appliances using the GUI

Before launching a *Complex Appliance*, make sure that you have a reservation for the appropriate node types and a key pair configured. Since most *Complex Appliances* will consist of multiple nodes, make sure you have set the *Minimum Number of Hosts* in your Lease. You will also need a *Template* file or the URL for a *Template* file. Templates for Chameleon-supported complex appliances can be found on Trovi by filtering for the **appliance** tag. At CHI@TACC site or CHI@UC site, go to *Project* > *Orchestration* > *Stacks* use the navigation side bar.

Tip

You can find complex appliance templates on Trovi by filtering for the **appliance** tag.

1. Go to Trovi, filter by the **appliance** tag, and identify the appliance you want to launch. Click on it to open its details page.
2. Download the template file or copy its URL from the artifact's detail page, then use it when launching a stack.

19.2.1 Launching a complex appliance

To launch a stack, click the *Launch Stack* button in the upper right of the *Stacks* page. Then follow the steps:

1. Start setting up a *Template* by choosing a *Template Source* in the dropdown. You may either select the *File* option as *Template Source* and upload the *Template* file, or select the *URL* option and provide the URL of the *Template* file.

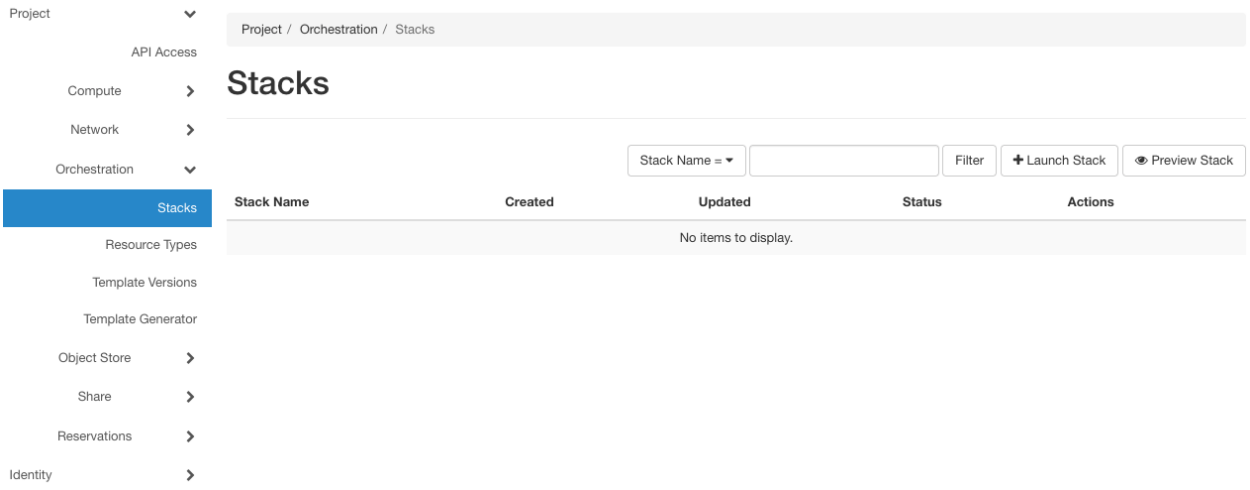


Fig. 1: The Stacks page

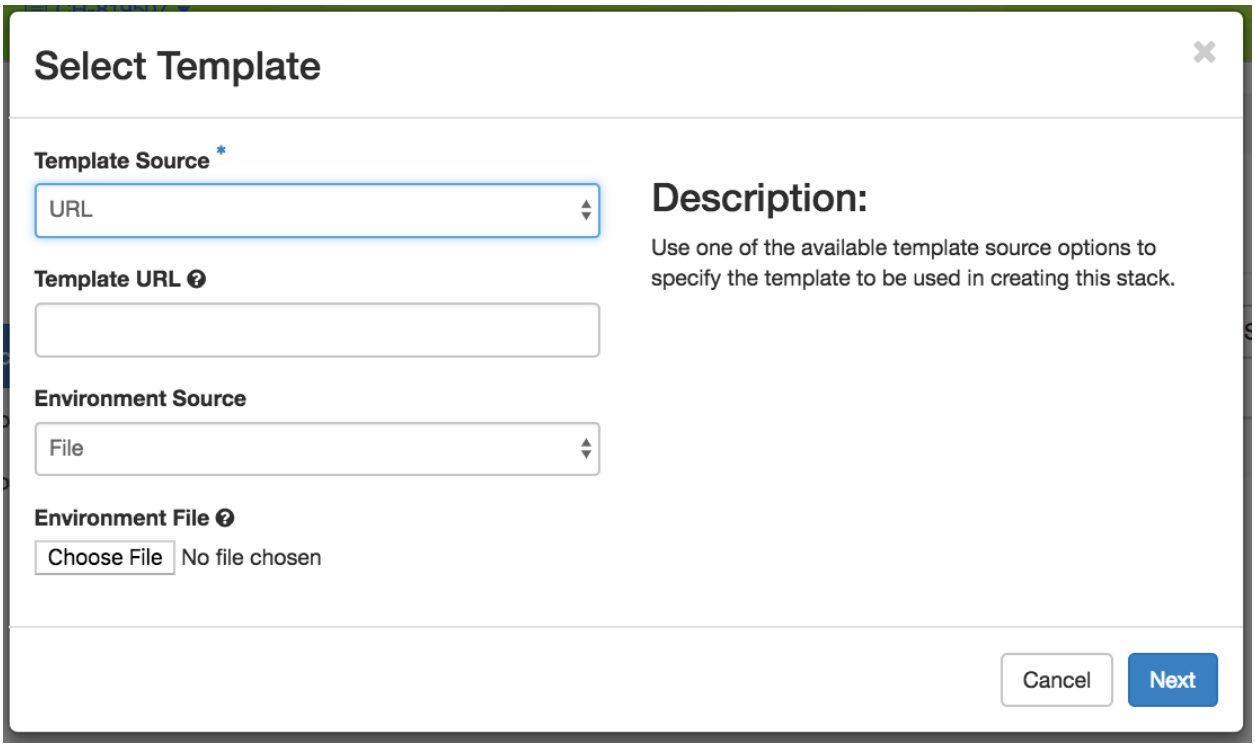


Fig. 2: The Select Template step

> **Important**
>
> **Do not** change the environment source settings!

2. Once you have provided a Template, click the *Next* button. Chameleon will validate the Template file and proceed to the *Launch Stack* step.

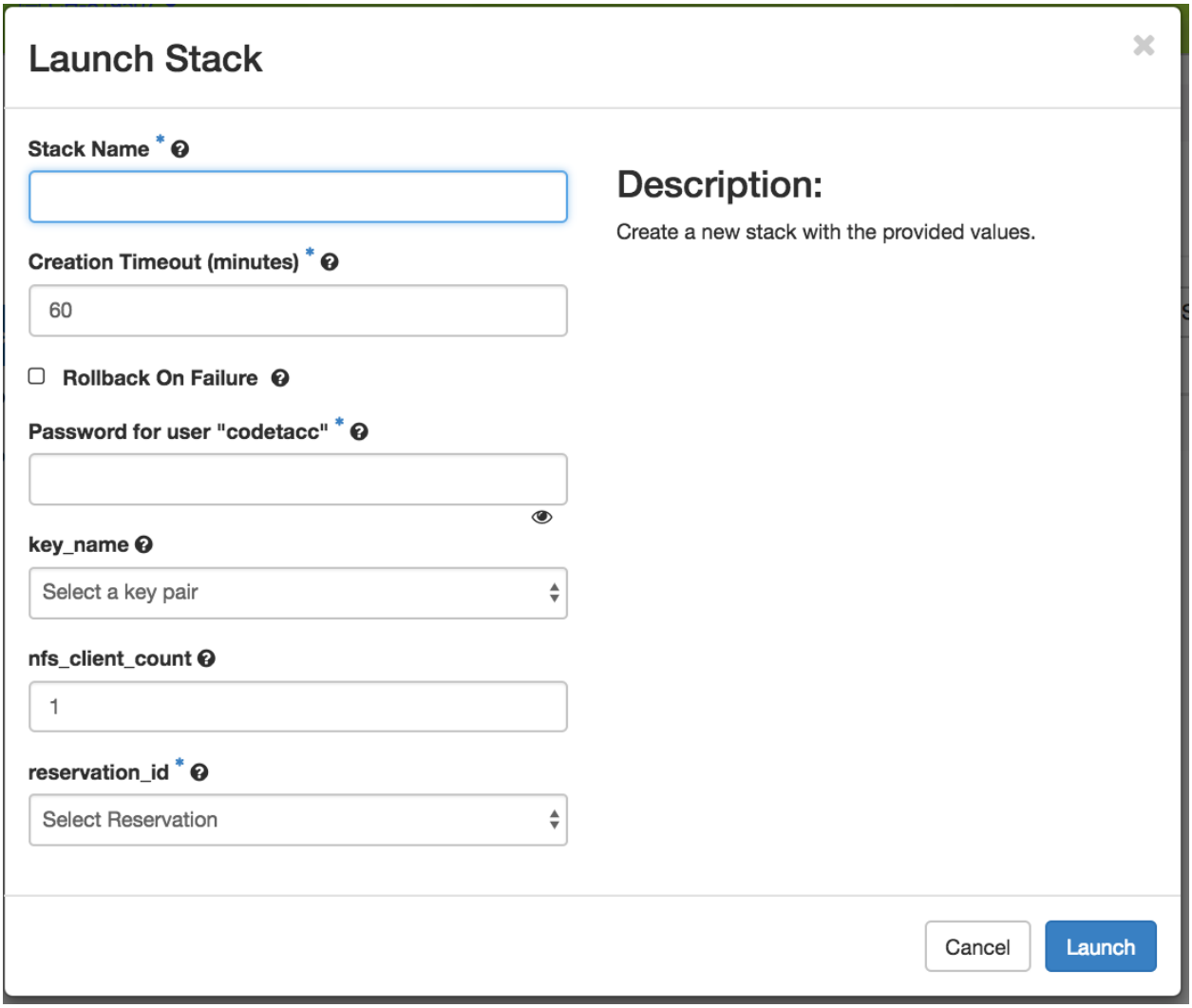


Fig. 3: The Launch Stack step

3. Choose a name for your stack. Ignore the "Creation Timeout" and "Rollback On Failure" settings. You also need to enter your Chameleon password. Then, you need to select a value for the parameters of the template. Finally, click the *Launch* button.
4. Your stack should be in status "Create In Progress" for several minutes while it first launches the server instance, followed by the client instances. It will then move to the status "Create Complete".

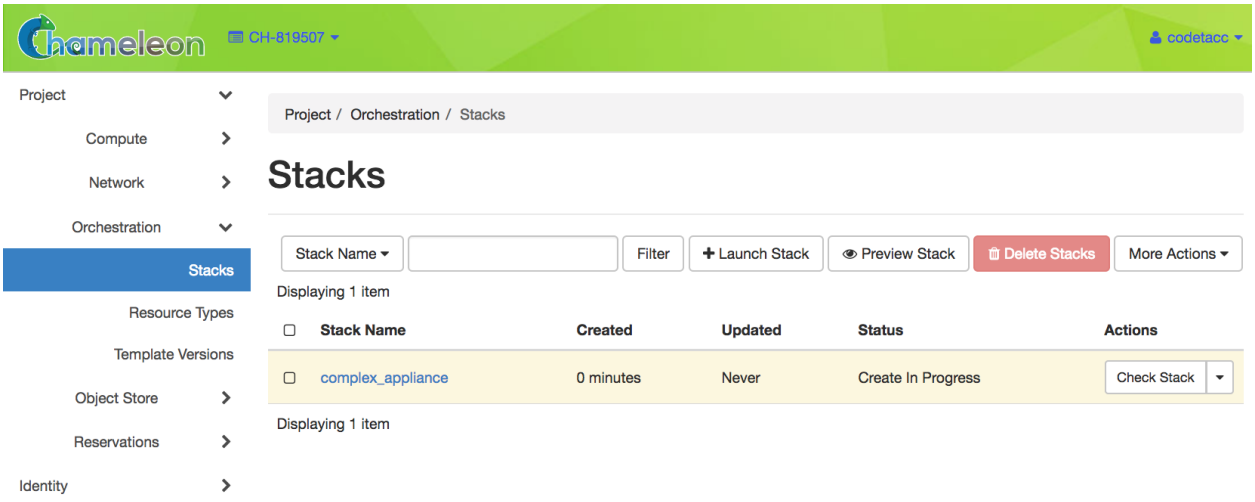


Fig. 4: A Complex Appliance with the Create in Progress status

19.2.2 Monitoring a complex appliance

To monitor and get more details about your *Complex Appliance*, click on it in the *Stacks* page.

- The *Topology* tab displays a topology graph of the stack. The rack of machine represents the client instance group. The server's floating IP (the public IP assigned to a resource) is represented by an IP in a circle; while an IP in a circle is also used to represent the association of the IP with the server instance (not the greatest idea to use the same symbol for both the IP and the association – we agree but can't do much about it at the moment). Blow off some steam by dragging the visualization across the screen, it can be rather fun!

> **Note**
>
> Blinking nodes indicates that they are still provisioning.

- The *Overview* tab displays various parameters, including the *ID* of the stack and *Outputs* such as IP addresses assigned to each node. If you have a floating IP associated to the server, you can now `ssh` to the server using the floating IP just as you do with regular instances. The client may not have a floating IP attached to it, but you can connect to it via the server node with the client's private IP.

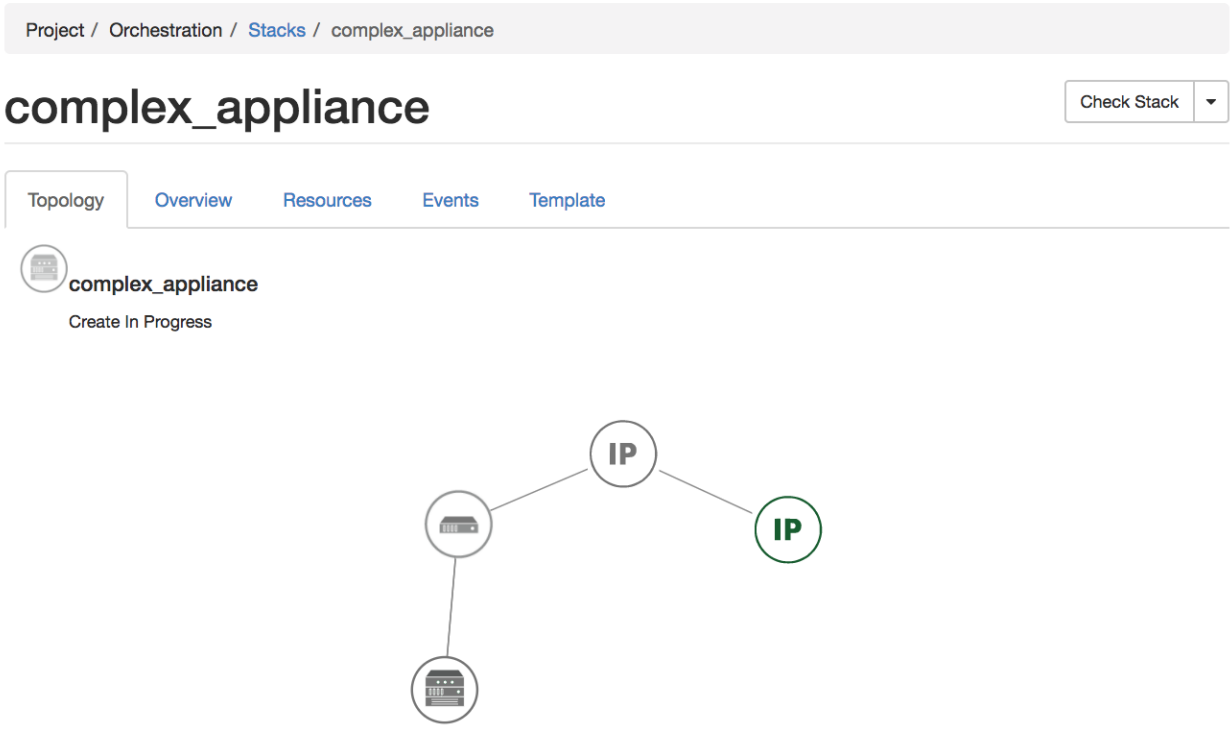


Fig. 5: The Topology tab

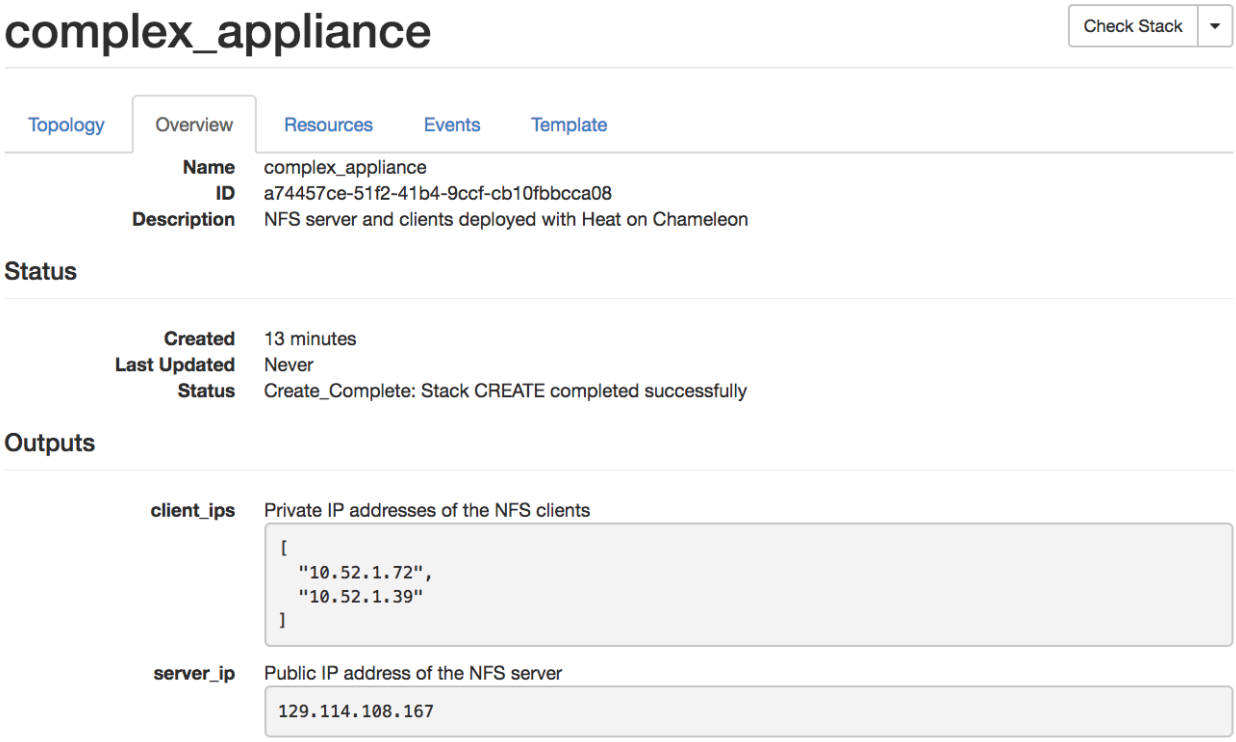


Fig. 6: The Overview tab

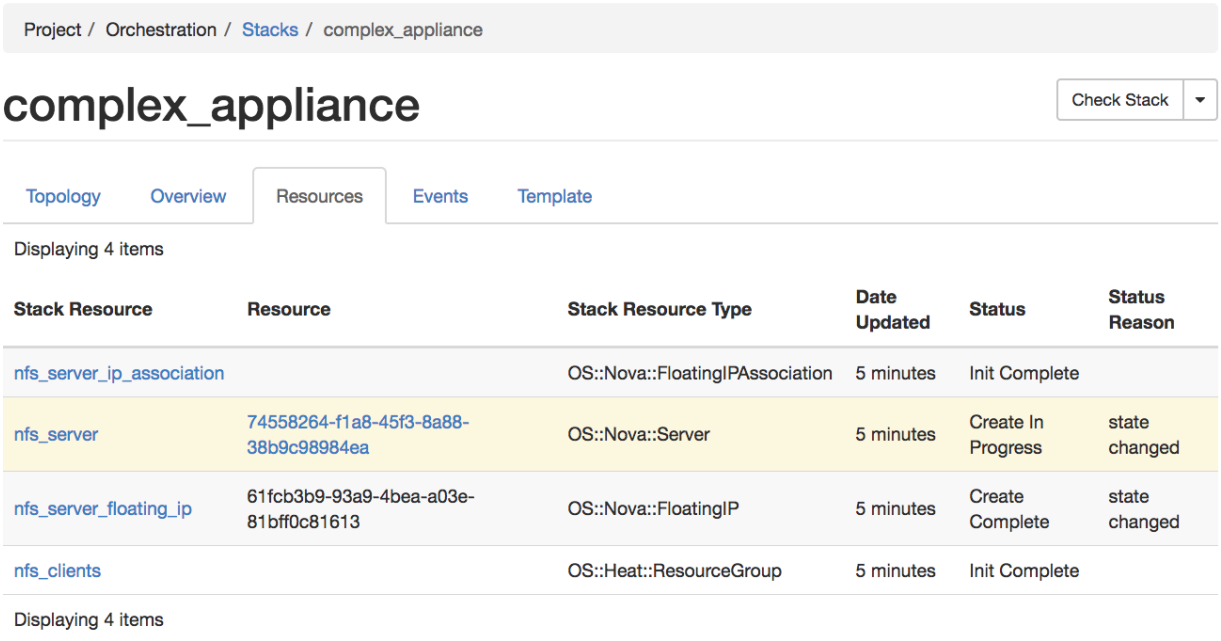


Fig. 7: The Resources tab

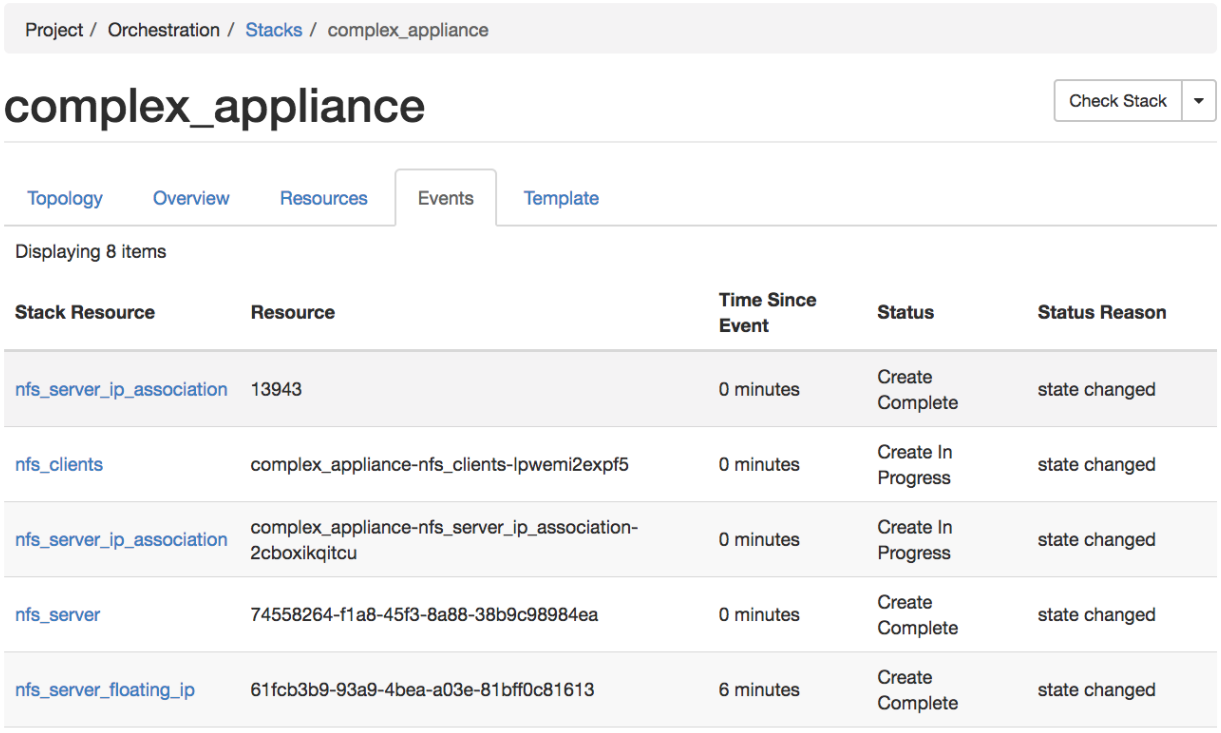


Fig. 8: The Events tab

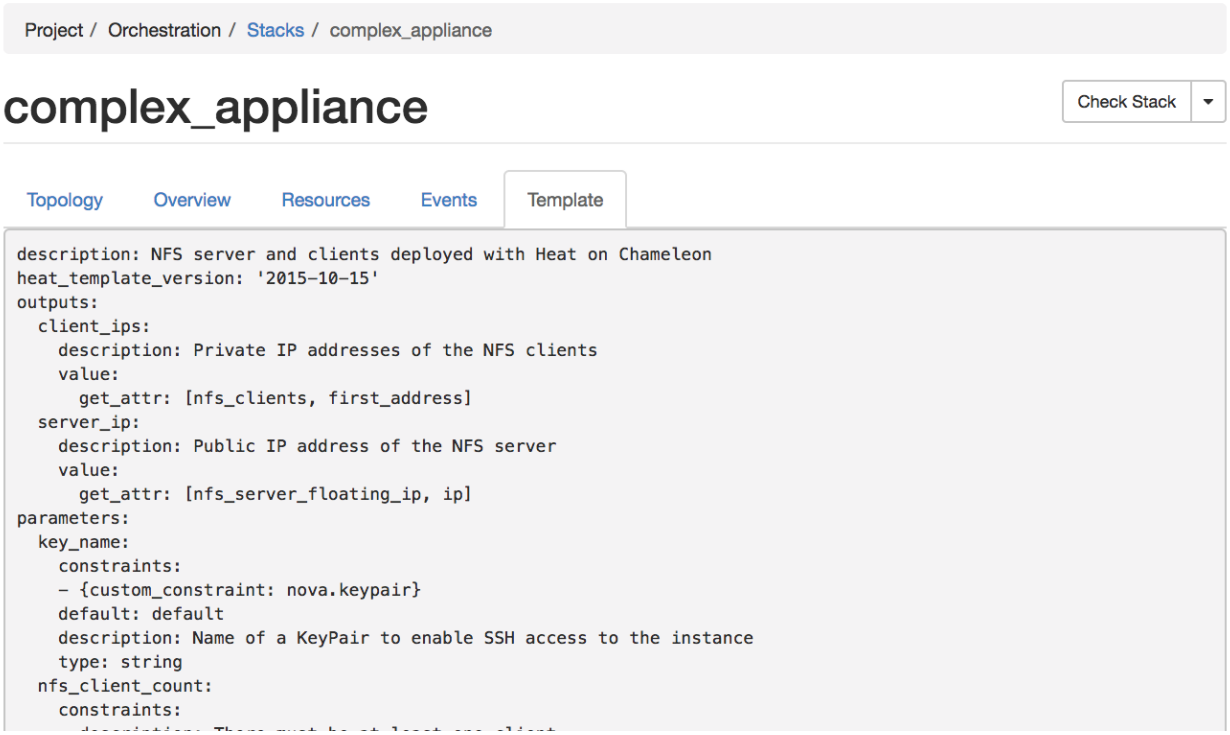


Fig. 9: The Template tab

> **Tip**
>
> To talk to the client without an associated floating IP, connect to the server with `ssh -A` to enable the SSH agent forwarding after loading your key to your SSH agent with `ssh-add <path-to-your-key>`.

- Under the *Resources* tab you will see the resources of the stack (the server, clients, server's public/floating IP, and its the association) and information about them.
- In the *Events* tab you will see information about the history of the deployment so far.
- In *Template* tab, you will see the template that was used to deploy this stack.

19.2.3 Deleting a complex appliance

To delete a *Complex Appliance*, select it in the *Stacks* page and click the *Delete Stacks* button. This will delete all resources of the stack, such as nodes and floating IP addresses.

19.3 Managing complex appliances using the CLI

> **Tip**
>
> Reading *Command Line Interface (CLI)* is highly recommended before continuing on the following sections.

In addition to *Installing the CLI*, you will need to install the `python-heatclient` package using the command:

```
pip install python-heatclient
```

Then, set up your environment for OpenStack command line usage, as described in *The OpenStack RC script*. You can get a list of your *Complex Appliances* in your project using the command:

```
openstack stack list
```

```
+--------------------------------------+--------------+------------------+-------------
↪---------+----------------------+
| ID                                   | Stack Name   | Stack Status     | Creation␣
↪Time        | Updated Time         |
+--------------------------------------+--------------+------------------+-------------
↪---------+----------------------+
| e5df33b5-5282-4935-8097-973328ca71e5 | my_stack     | CREATE_COMPLETE  | 2018-01-
↪23T22:45:12Z | None                 |
+--------------------------------------+--------------+------------------+-------------
↪---------+----------------------+
```

19.3.1 Launching a complex appliance

To launch a *Complex Appliance* using *Template*, run the command on your local machine:

```
openstack stack create --template <template_file> --parameter <parameter>=<value> <stack_
↪name>
```

Provide the path to and the name of the *Template* file in your local file system via the `template` switch. The `<stack_name>` is the name of the *Complex Appliance*. In addition, you may provide the parameters required in the *Template* file with their values by `parameter` switch. For example, the NFS Server Template (find it on Trovi by filtering for the **appliance** tag) lists the following `parameters` section:

```
parameters:
  nfs_client_count:
    type: number
    description: Number of NFS client instances
    default: 1
    constraints:
      - range: { min: 1 }
        description: There must be at least one client.
  key_name:
    type: string
    description: Name of a KeyPair to enable SSH access to the instance
    default: default
    constraints:
    - custom_constraint: nova.keypair
  reservation_id:
    type: string
    description: ID of the Blazar reservation to use for launching instances.
    constraints:
    - custom_constraint: blazar.reservation
```

Therefore, in order to use this *Template*, you must provide values for `nfs_client_count`, `key_name` and `reservation_id`.

19.3.2 Monitoring a complex appliance

You can get details about your *Complex Appliance*, such as *Outputs*, *Events* and *Resources*, via the CLI. You will need the *UUID* of the *Complex Appliance*.

Tip

To get the *UUID* of your *Complex Appliance*, use the *Stacks* page on the GUI or retrieve it by `openstack stack list` command.

- To view the *Outputs*, run:

```
openstack stack output list <uuid>
```

For example, the list of the outputs for the NFS Share stack is:

```
+------------+-----------------------------------------+
| output_key | description                             |
+------------+-----------------------------------------+
| client_ips | Private IP addresses of the NFS clients |
| server_ip  | Public IP address of the NFS server     |
+------------+-----------------------------------------+
```

You can get more details about the outputs by using the following command:

```
openstack stack output show --all <uuid>
```

- To view the *Events*, run:

```
openstack stack event list <uuid>
```

- To view the *Resources*, run:

```
openstack stack resource list <uuid>
```

Your output may look like this:

```
+--------------------------+--------------------------------------+---------------
↪-----------------+----------------+----------------------+
| resource_name            | physical_resource_id                 | resource_type ␣
↪                 | resource_status | updated_time         |
+--------------------------+--------------------------------------+---------------
↪-----------------+----------------+----------------------+
| nfs_server_ip_association |                                      |␣
↪OS::Neutron::FloatingIPAssociation | INIT_COMPLETE   | 2018-03-19T18:38:05Z |
| nfs_server               | 0ab0169c-f762-4d27-8724-b359caa50f1f |␣
↪OS::Nova::Server               | CREATE_FAILED   | 2018-03-19T18:38:05Z |
| nfs_server_floating_ip   | ecb391f8-4653-43a6-b2c6-bb93a6d89115 |␣
↪OS::Nova::FloatingIP           | CREATE_COMPLETE | 2018-03-19T18:38:05Z |
| nfs_clients              |                                      |␣
↪OS::Heat::ResourceGroup        | INIT_COMPLETE   | 2018-03-19T18:38:05Z |
+--------------------------+--------------------------------------+---------------
↪-----------------+----------------+----------------------+
```

Then, you may retrieve information about a specific resource using the command:

```
openstack stack resource show <stack_uuid> <resource_name>
```

19.3.3 Deleting a complex appliance

Use the following command to delete a stack:

```
openstack stack delete <uuid>
```

It will remove all the resources attached to the stack.

19.4 Heat orchestration templates

A *Heat Orchestration Template* is a YAML file that specifies how resources are used and configured in a *Complex Appliance*.

19.4.1 A case example: NFS share

Let's look at the NFS Share Template (available on Trovi). The NFS share appliance deploys:

- An NFS server instance, that exports the directory `/exports/example` to any instance running on Chameleon bare metal,
- One or several NFS client instances, which configure `/etc/fstab` to mount this NFS share to `/mnt` (and can subsequently read from and write to it).

This template is reproduced further below, and includes inline comments starting with the # character. There are three main sections:

- resources
- parameters
- outputs

The `resources` section is the most important part of the template: it defines which OpenStack *Resources* to create and configure. Inside this section you can see four resources defined:

- `nfs_server_floating_ip`: creates a *Floating IP* on the `ext-net` public network. It is not attached to any instance yet.
- `nfs_server`: creates the NFS server instance (an instance is defined with the type `OS::Nova::Server` in *Heat*). It is a bare metal instance (`flavor: baremetal`) using the `CC-CentOS7` image and connected to the private network named `sharednet1`. We set the key pair to use the value of the parameter defined earlier, using the `get_param` function. Similarly, the reservation to use is passed to the scheduler. Finally, a `user_data` script is given to the instance, which configures it as an NFS server exporting `/exports/example` to Chameleon instances.
- `nfs_server_ip_association`: associates the floating IP created earlier with the NFS server instance.
- `nfs_clients`: defines a resource group containing instance configured to be NFS clients and mount the directory exported by the NFS server defined earlier. The IP of the NFS server is gathered using the `get_attr` function, and placed into `user_data` using the `str_replace` function.

Once a Resource has been specified, you may provide it as a value for another Resource's property using the `get_resource` function.

The `parameters` section defines inputs to be used on *Complex Appliance* launch. Parameters all have the same data structure: each one has a name (`key_name` or `reservation_id` in this case), a data type (`number` or `string`), a comment field called `description`, optionally a `default value`, and a list of `constraints` (in this case only one per parameter). Constraints tell *Heat* to match a parameter to a specific type of OpenStack resource. *Complex appliances* on Chameleon require users to customize at least the key pair name and reservation ID, and will generally provide additional parameters to customize other properties of the cluster, such as its size, as in this example. The values of Parameters can be used in the `resources` section using the `get_param` function.

The `outputs` section defines what values are returned to the user. *Outputs* are declared similarly to *Parameters*: they each have a name, an optional description, and a value. They allow to return information from the stack to the user. You may use the `get_attr` function to retrieve a resource's attribute for output.

19.4.2 Heat template customization

Customizing an existing template is a good way to start developing your own. We will use a simpler template than the previous example to start with: the Hello World complex appliance, which you can find on Trovi.

First, delete the stack you launched, because we will need all three nodes to be free. To do this, go back to the *Project* > *Orchestration* > *Stacks* page, select your stack, and then click on the *Delete Stacks* button. You will be asked to confirm, so click on the *Delete Stacks* button.

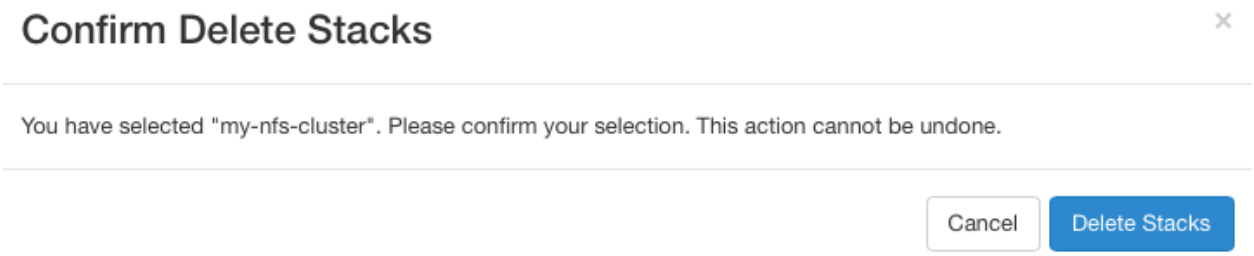


Fig. 10: Confirm deleting stack dialog

The template for the Hello World complex appliance is reproduced below. It is similar to the NFS share appliance, except that it deploys only a single client. You can see that it has four resources defined:

- `nfs_server_floating_ip`
- `nfs_server`
- `nfs_server_ip_association`
- `nfs_client`

The `nfs_client` instance mounts the NFS directory shared by the `nfs_server` instance, just like in our earlier example.

```
# This describes what is deployed by this template.
description: NFS server and client deployed with Heat on Chameleon

# This defines the minimum Heat version required by this template.
heat_template_version: 2015-10-15

# The resources section defines what OpenStack resources are to be deployed and
# how they should be configured.
resources:
  nfs_server_floating_ip:
    type: OS::Nova::FloatingIP
    properties:
      pool: ext-net

  nfs_server:
    type: OS::Nova::Server
```

(continues on next page)

(continued from previous page)

```
    properties:
      flavor: baremetal
      image: CC-CentOS7
      key_name: { get_param: key_name }
      networks:
         - network: sharednet1
      scheduler_hints: { reservation: { get_param: reservation_id } }
      user_data: |
        #!/bin/bash
        yum install -y nfs-utils
        mkdir -p /exports/example
        chown -R cc:cc /exports
        echo '/exports/example 10.140.80.0/22(rw,async) 10.40.0.0/23(rw,async)' >> /etc/
↪exports
        systemctl enable rpcbind && systemctl start rpcbind
        systemctl enable nfs-server && systemctl start nfs-server

  nfs_server_ip_association:
    type: OS::Neutron::FloatingIPAssociation
    properties:
      floatingip_id: {get_resource: nfs_server_floating_ip}
      port_id: {get_attr: [nfs_server, addresses, sharednet1, 0, port]}

  nfs_client:
    type: OS::Nova::Server
    properties:
      flavor: baremetal
      image: CC-CentOS7
      key_name: { get_param: key_name }
      networks:
         - network: sharednet1
      scheduler_hints: { reservation: { get_param: reservation_id } }
      user_data:
        str_replace:
          template: |
            #!/bin/bash
            yum install -y nfs-utils
            echo "$nfs_server_ip:/exports/example    /mnt/    nfs" > /etc/fstab
            mount -a
          params:
            $nfs_server_ip: { get_attr: [nfs_server, first_address] }

# The parameters section gathers configuration from the user.
parameters:
  key_name:
    type: string
    description: Name of a KeyPair to enable SSH access to the instance
    default: default
    constraints:
    - custom_constraint: nova.keypair
  reservation_id:
    type: string
```

(continues on next page)

(continued from previous page)

```
    description: ID of the Blazar reservation to use for launching instances.
    constraints:
    - custom_constraint: blazar.reservation
```

Copy the template from the artifact's detail page on Trovi to your local machine, and open it in your favorite text editor.

We will customize the template to add a second NFS client by creating a new resource called `another_nfs_client`. Add the following text to your template inside the resources section. Make sure to respect the level of indentation, which is important in YAML.

```
another_nfs_client:
  type: OS::Nova::Server
  properties:
    flavor: baremetal
    image: CC-CentOS7
    key_name: { get_param: key_name }
    networks:
       - network: sharednet1
    scheduler_hints: { reservation: { get_param: reservation_id } }
    user_data:
      str_replace:
        template: |
          #!/bin/bash
          yum install -y nfs-utils
          echo "$nfs_server_ip:/exports/example    /mnt/    nfs" > /etc/fstab
          mount -a
        params:
          $nfs_server_ip: { get_attr: [nfs_server, first_address] }
```

Now, launch a new stack with this template. Since the customized template is only on your computer and cannot be addressed by a URL, use the *Direct Input* method instead and copy/paste the content of the customized template. The resulting topology view is shown below: as you can see, the two client instances are shown separately since each one is defined as a separate resource in the template.

You may have realized already that while adding just one additional client instance was easy, launching more of them would require to copy / paste blocks of YAML many times while ensuring that the total count is correct. This would be easy to get wrong, especially when dealing with tens or hundreds of instances.

So instead, we leverage another construct from *Heat*: resource groups. Resource groups allow to define one kind of resource and request it to be created any number of times.

Remove the `nfs_client` and `another_client` resources from your customized template, and replace them with the following:

```
nfs_clients:
  type: OS::Heat::ResourceGroup
  properties:
    count: 2
    resource_def:
      type: OS::Nova::Server
      properties:
        flavor: baremetal
        image: CC-CentOS7
```

(continues on next page)

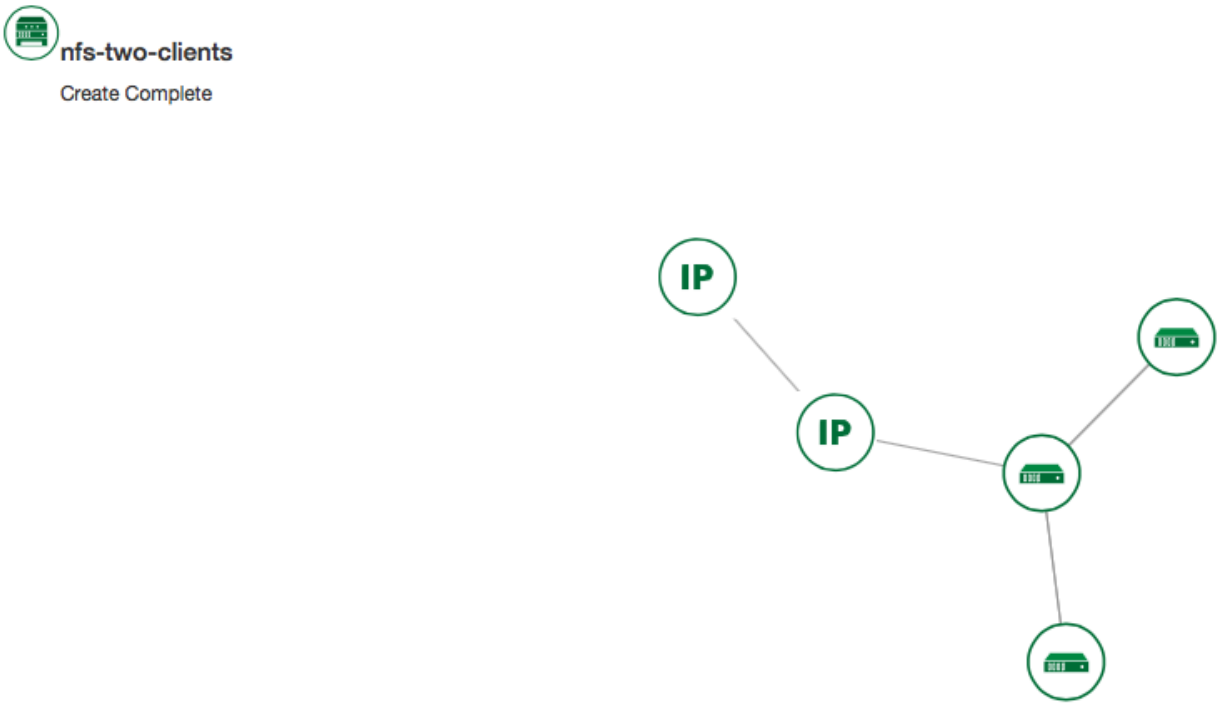


Fig. 11: Topology of the customized Hello World Appliance

(continued from previous page)

```
      key_name: { get_param: key_name }
      networks:
         - network: sharednet1
      scheduler_hints: { reservation: { get_param: reservation_id } }
      user_data:
        str_replace:
          template: |
            #!/bin/bash
            yum install -y nfs-utils
            echo "$nfs_server_ip:/exports/example    /mnt/    nfs" > /etc/fstab
            mount -a
          params:
            $nfs_server_ip: { get_attr: [nfs_server, first_address] }
```

A resource group is configured with a properties field, containing the definition of the resource to launch (`resource_def`) and the number of resources to launch (`count`). Once launched, you will notice that the topology view groups all client instances under a single *Resource Group* icon. We use the same `resource_def` than when defining separate instances earlier.

Another way we can customize this template is by adding outputs to the template. Outputs allow a *Heat* template to return data to the user. This can be useful to return values like IP addresses or credentials that the user must know to use the system.

We will create an output returning the floating IP address used by the NFS server. We define an outputs section, and one output with the name `server_ip` and a description. The value of the output is gathered using the `get_attr` function which obtains the IP address of the server instance.

```
outputs:
  server_ip:
    description: Public IP address of the NFS server
    value: { get_attr: [nfs_server_floating_ip, ip] }
```

You can get outputs in the *Overview* tab of the *Stack Details* page. If you want to use the command line, install `python-heatclient` and use the `heat output-list` and `heat output-show` commands, or get a full list in the information returned by `heat stack-show`.

Multiple outputs can be defined in the outputs section. Each of them needs to have a unique name. For example, we can add another output to list the private IPs assigned to client instances:

```
client_ips:
  description: Private IP addresses of the NFS clients
  value: { get_attr: [nfs_clients, first_address] }
```

The image below shows the resulting outputs as viewed from the GUI. Of course IP addresses will be specific to each deployment.

Fig. 12: The Outputs of customized Hello World appliance

Finally, we can add a new parameter to replace the hard-coded number of client instances by a value passed to the template. Add the following text to the parameters section:

```
nfs_client_count:
  type: number
  description: Number of NFS client instances
  default: 1
  constraints:
    - range: { min: 1 }
      description: There must be at least one client.
```

Inside the resource group definition, change `count: 2` to `count: { get_param: nfs_client_count }` to retrieve and use the parameter we just defined. When you launch this template, you will see that an additional parameter allows you to define the number of client instances, like in the NFS share appliance.

At this stage, we have fully recreated the *NFS share* appliance starting from the *Hello World* one! The next section will explain how to write a new template from scratch.

19.4.3 Writing a new template

You may want to write a whole new template, rather than customizing an existing one. Each template should follow the same layout and be composed of the following sections:

- Heat template version
- Description
- Resources
- Parameters
- Outputs

Heat template version

Each Heat template has to include the `heat_template_version` key with a valid version of HOT (Heat Orchestration Template). Chameleon bare metal supports any HOT version up to **2015-10-15**, which corresponds to OpenStack Liberty. The Heat documentation lists all available versions and their features. We recommended that you always use the latest Chameleon supported version to have access to all supported features:

```
heat_template_version:  2015-10-15
```

Description

While not mandatory, it is good practice to describe what is deployed and configured by your template. It can be on a single line:

```
description: This describes what this Heat template deploys on Chameleon.
```

If a longer description is needed, you can provide multi-line text in YAML, for example:

```
description: >
  This describes what this Heat
  template deploys on Chameleon.
```

Resources

The resources section is required and must contain at least one resource definition. A complete list of resources types known to Heat is available.

However, only a subset of them are supported by Chameleon, and some are limited to administrative use. We recommend that you only use:

- OS::Glance::Image
- OS::Heat::ResourceGroup
- OS::Heat::SoftwareConfig
- OS::Heat::SoftwareDeployment
- OS::Heat::SoftwareDeploymentGroup
- OS::Neutron::FloatingIP
- OS::Neutron::FloatingIPAssociation
- OS::Neutron::Port (advanced users only)
- OS::Nova::Keypair
- OS::Nova::Server

If you know of another resource that you would like to use and think it should be supported by the OpenStack services on Chameleon bare metal, let us know via our Help Desk.

Parameters

Parameters allow users to customize the template with necessary or optional values. For example, they can customize which Chameleon appliance they want to deploy, or which key pair to install. Default values can be provided with the `default` key, as well as constraints to ensure that only valid OpenStack resources can be selected. For example, `custom_constraint: glance.image` restricts the image selection to an available OpenStack image, while providing a pre-filled selection box in the GUI. More details about constraints are available in the *Heat* documentation.

Outputs

Outputs allow template to give information from the deployment to users. This can include usernames, passwords, IP addresses, hostnames, paths, etc. The outputs declaration is using the following format:

```
outputs:
  first_output_name:
    description: Description of the first output
    value: first_output_value
  second_output_name:
    description: Description of the second output
    value: second_output_value
```

Generally values will be calls to `get_attr`, `get_param`, or some other function to get information from parameters or resources deployed by the template and return them in the proper format to the user.

19.4.4 Reserved networks and floating IPs

Chameleon's reservation service allows users to reserve VLAN segments and floating ips. In order to make use of these reserved resources in a (HOT) template, follow the guidelines below. For more information on VLAN and floating ip reservations, see documentaiton on *Creating a lease to reserve a VLAN segment* and *Creating a lease to reserve floating IPs*

When you reserve a VLAN segment via blazar, it will automatically create a network for you. However, this network is not usable in your template unless a subnet and router have been associated with the network. Once this is done, you can simply add the network name as the network parameter for your server as you would `sharednet1`. The below cli commands provides an example of how to complete the configuration for your reserved network.

```
openstack subnet create --subnet-range 192.168.100.0/24 \
    --allocation-pool start=192.168.100.100,end=192.168.100.108 \
    --dns-nameserver 8.8.8.8 --dhcp \
    --network <my_reserved_network_name> \
    my_subnet_name
openstack router create my_router_name
openstack router add subnet my_router_name my_subnet_name
openstack router set --external-gateway public my_router_name
```

For reserved floating ips, you need to associate the floating ip with a server using the `OS::Neutron::FloatingIPAssociation` object type. Many of our older complex appliance templates use the `OS::Nova::FloatingIPAssociation` object, but this has since been deprecated. See example below for proper usage:

```
my_server_ip_association:
  type: OS::Neutron::FloatingIPAssociation
```

(continues on next page)

(continued from previous page)

```
  properties:
    floatingip_id: <my_reserved_floating_ip_uuid>
    port_id: {get_attr: [my_server, addresses, <my_network_name>, 0, port]}
```

If you are having trouble finding the `uuid` of the floating ip address then the below command will help you.

```
openstack floating ip list -c ID -c "Floating IP Address" -c Tags --long
```

The output should look like the sample output below with the *uuid* listed under the *ID* column. You can check your lease in the reservation section of the GUI to find the *reservation id* associated with the floating ip in the *Tags* section of the output.

```
+--------------------------------------+---------------------+--------------------------
↪----------------------------------------+
| ID                                   | Floating IP Address | Tags                    ␣
↪                                      |
+--------------------------------------+---------------------+--------------------------
↪----------------------------------------+
| 0fe31fad-60ac-462f-bb6c-4d40c1506621 | 192.5.87.206        | [u'reservation:d90ad917-
↪300a-4cf7-a836-083534244f56', u'blazar'] |
| 92a347a9-31a5-43c1-80e2-9cdb38ebf66f | 192.5.87.224        | [u'reservation:5f470c97-
↪0166-4934-a813-509b743e2d62', u'blazar'] |
| c8480d67-533d-4f55-a197-8271da6d9344 | 192.5.87.71         | []                      ␣
↪                                      |
+--------------------------------------+---------------------+--------------------------
↪----------------------------------------+
```

19.5 Sharing complex appliances

If you have written your own *Complex Appliance* or substantially customized an existing one, we would love if you shared them with our user community! Appliances are now published through Trovi — the Appliance Catalog no longer accepts new submissions.

To publish a new appliance, first create or find your artifact in Trovi, then open its **Edit** menu. From there you can add the **appliance** tag and provide a name, description, heat template, and other relevant metadata. Once published, your appliance will be discoverable by other Chameleon users when they filter Trovi by the **appliance** tag.

> **Note**
>
> The old Appliance Catalog is deprecated and no longer used. All appliances — images, heat templates, and other artifacts — are now stored and shared exclusively through Trovi.

19.6 Advanced topics

The previous examples have all used `user_data` scripts to provide instances with contextualization information. While it is easy to use, this contextualization method has a major drawback: because it is given to the instance as part of its launch request, it cannot use any context information that is not yet known at this time. In practice, this means that in a client-server deployment, only one of these pattern will be possible:

- The server has to be deployed first, and once it is deployed, the clients can be launched and contextualized with information from the server. The server won't know about the clients unless there is a mechanism (not managed

by *Heat*) for the client to contact the server.

- The clients have to be deployed first, and once they are deployed, the server can be launched and contextualized with information from the clients. The clients won't know about the server unless there is a mechanism (not managed by *Heat*) for the server to contact the clients.

This limitation was already apparent in our NFS share appliance (available on Trovi): this is why the server instance exports the file system to all bare metal instances on Chameleon, because it doesn't know which specific IP addresses are allocated to the clients.

This limitation is even more important if the deployment is not hierarchical, i.e. all instances need to know about all others. For example, a cluster with IP and hostnames populated in `/etc/hosts` required each instance to be known by every other instance.

To solve this problem we recommend performing contextualization after the instances have booted. Rather than passing all necessary context information at launch time, which may be incomplete or not yet availaable, you can use configuration management tools such as Ansible, to orchestrate all-to-all information exchange between the nodes.

Once all instances have booted and are online, the general flow is:

- Gather required information from all instances (IP addresses, hostnames, etc.)
- Generate configuration files (e.g., /etc/hosts) and secrets (e.g., SSH keys), etc.
- Push configuration files and secrets to all instances.

A tool like Ansible can be run directly from your laptop, or from another environment like a dedicated server or Jupyter notebook. By separating the deployment of instances (using Heat, or python-chi) from their contextualization (using Ansible or another tool), any necessary all-to-all information exchange can be performed.

One specific example of this approach is our MPI Trovi artifact.

This artifact contains the necessary python-chi and Ansible scripts to deploy a cluster of instances capable of running an MPI application.

CHAPTER

TWENTY

OBJECT STORE

Chameleon provides an object store service through the OpenStack Swift interface. It is intended to be used for storing and retrieving data used during experiments, such as input files needed for your applications, or results produced by your experiments.

Note

Objects can also be managed programmatically via the chi.storage module in python-chi — see our *Jupyter and python-chi guide* for an introduction.

Hint

Chameleon object store service is currently backed by a Ceph cluster with more than 2.1 PB of capacity. The data is replicated, keeping two copies of each object, effectively providing over 1 PB of storage available to users. This storage capacity will increase as the project goes on. The replication should provide good availability in case of hardware failures. However, all copies are kept within the same data center and are not backed up on a separate system; if you feel that this does not provide sufficient reliability in your case, you should consider backing up really critical data externally.

20.1 Availability

You can access the *Object Store* from instances running on CHI@TACC and CHI@UC. Each region has its own store, meaning that objects uploaded to one are not visible to the other. In general you should use the store local to the region where your instances are running for the best performance. To make it easier for you to use the *Object Store* client, we installed it in all appliances supported by Chameleon. Additionally, you can also access the *Object Store* from the CHI@TACC or CHI@UC GUIs under the *Object Store* panel.

Hint

KVM@TACC users can access the TACC store using `ccauth` to generate multi-site credentials. See *Accessing object store from KVM instances* for step-by-step instructions.

20.2 Objects and containers

Objects are equivalent to individual files. They are stored in *Containers*, which are data structures that can contain multiple *Objects*. When uploading *Objects*, they must be stored inside of *Containers*. You may perform operations on

individual *Objects* inside Containers, such as downloading or deleting them. You may also work with entire *Containers* and perform operations such as downloading an entire *Container*.

20.2.1 Managing object store using the GUI

To access the *Object Store* using the GUI at a CHI site, use the navigation sidebar to go to *Project* > *Object Store* > *Containers*.

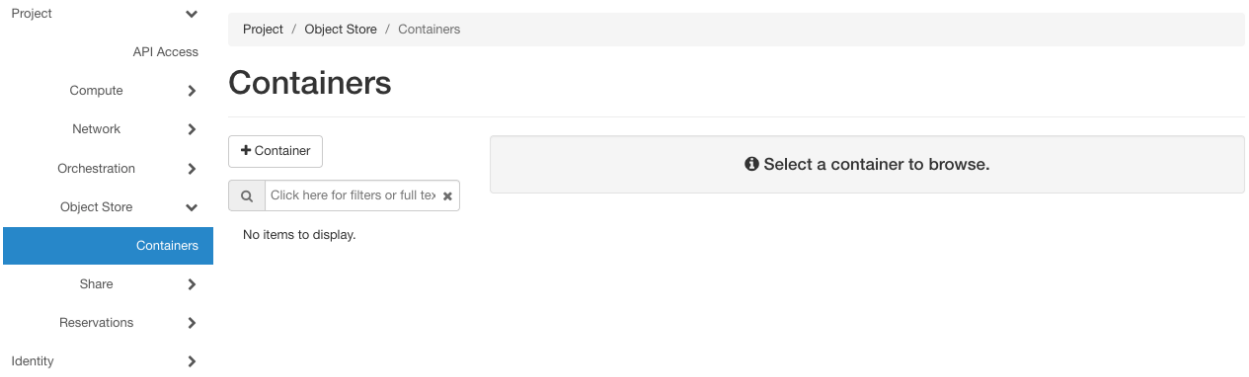


Fig. 1: The Containers page

Working with containers

To create a container, click the *+Container* button. This will open the *Create Container* dialog.

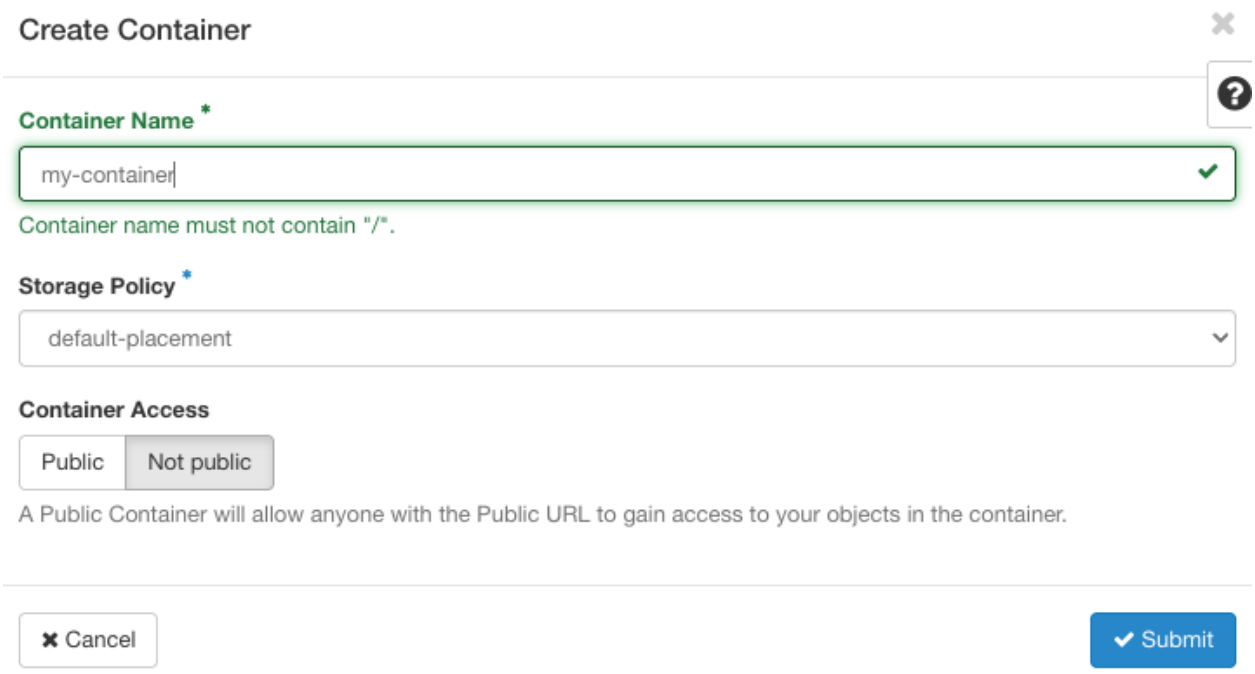


Fig. 2: The Create Container dialog

Choose a unique name of your container and set the visibility to either *Public* or *Not Public*. When you are finished, click the *Submit* button. You will see your new *Container* appear in the list on the *Containers* page.

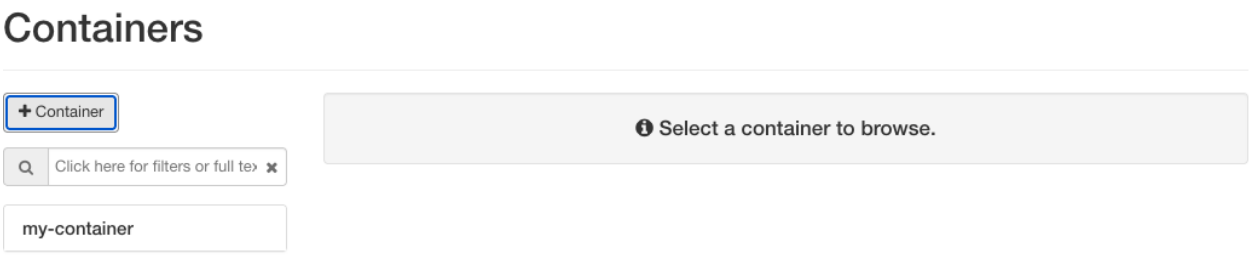


Fig. 3: The Container list

You may click on a *Container* to see the details and work with *Objects* belonging to it.

Attention

Downloading a container is not available from the GUI. Use the CLI to download containers.

You may delete a container by clicking the *Delete* icon in the upper right of the *Container Detail Panel*.

Fig. 4: The Delete Container button

Working with objects

To upload a local file to a container, click the button with the *Upload* symbol next to the search bar.

This will open the *Upload File* dialog.

Choose a file to upload from your local file system and give a name to the object.

Working with folders

If you wish to create a *Folder* within your *Container*, click the *+Folder* button and give a name to your folder in the *Create Folder* dialog.

Your new folder will appear in the *Container details*.

You may browse your folder and upload files to it by clicking on the folder.

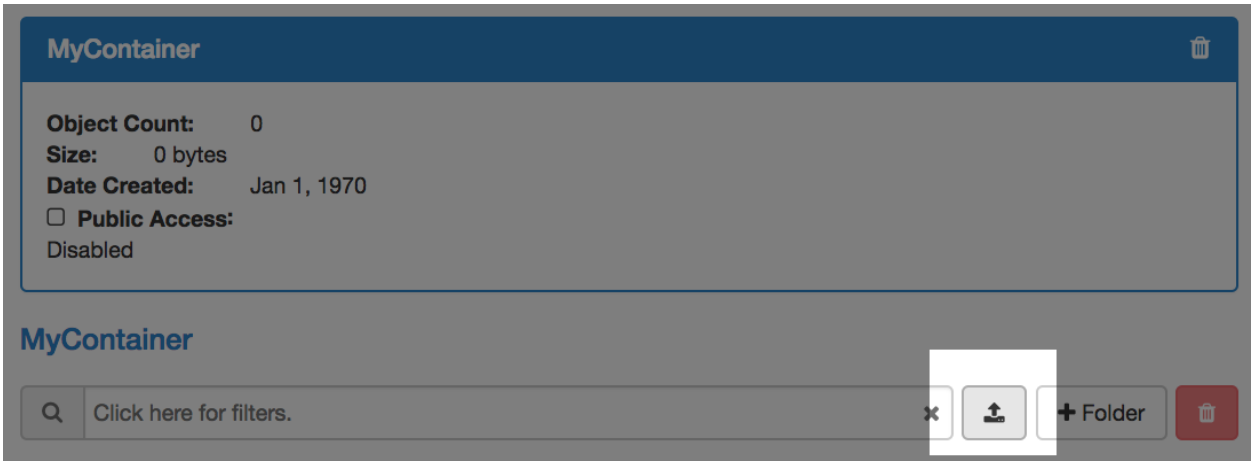


Fig. 5: The Upload button

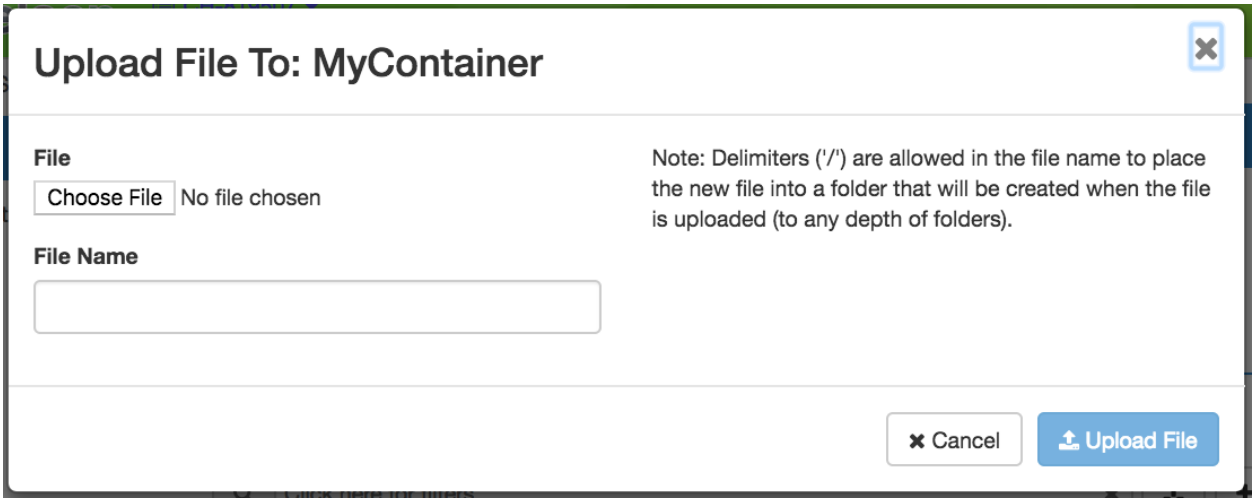


Fig. 6: The Upload File dialog

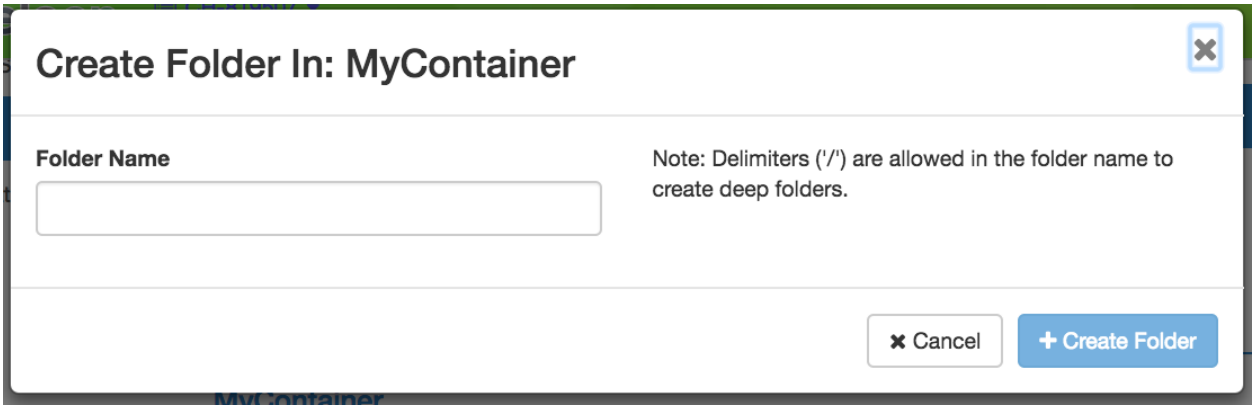


Fig. 7: The Create Folder dialog

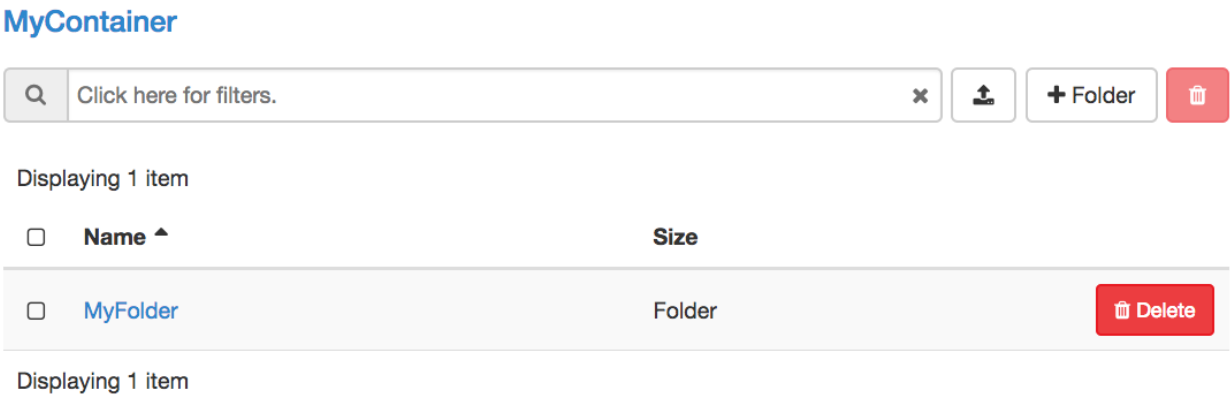


Fig. 8: A Container with a Folder

Fig. 9: A Folder within the Container

20.2.2 Managing object store using the CLI

> **Tip**
>
> Reading *Command Line Interface (CLI)* is highly recommended before continuing on the following sections.

In addition to *Installing the CLI*, you must also install `python-swiftclient` package:

```
pip install python-swiftclient
```

Then, you must set environment variables for your account and project using *The OpenStack RC script*.

Working with containers

To create a *Container*, use the following command:

```
openstack container create <container_name>
```

> **Tip**
>
> By default, the *Container* created using the above command will not be visible to the public.

To view all containers that belong to your project, run:

```
openstack container list
```

> **Tip**
>
> You may use `--prefix <prefix>` as a filter to list the containers whose name starts with `<prefix>`.

To see details of a container, use the command:

```
openstack container show <container_name>
```

To view a list of objects within a container, use the command:

```
openstack object list <container_name>
```

To download a container with all the objects belong to it, use the following command:

```
openstack container save <container_name>
```

To delete a container and wipe out all the objects belong to it, use the following command, and **be careful**!

```
openstack container delete --recursive <container_name>
```

Working with objects

You may upload a file from your local machine to a container using the following command:

```
openstack object create <container_name> <local_filename>
```

> **Tip**
>
> Optionally, you may name the object differently from it's original name in your local machine by using the `--name` parameter.

To delete an object from a container, run:

```
openstack object delete <container_name> <object_name>
```

If you wish to download an individual object directly from a container, use the command:

```
openstack object save <container_name> <object_name>
```

Large object support

The Swift CLI only supports objects up to 4GB. Larger objects are supported, provided they are uploaded in segments. This advanced functionality is only supported using a separate Swift interface. For a version compatible with Chameleon's authentication, you need *python-swiftclient >= 3.11.1*, and to generate and use an *Application Credential*

```
pip install "python-swiftclient>=3.11.1"
```

Instead of invoking commands via `openstack`, you will instead use the `swift` command, which supports a `--segment-size` parameter, specifying the segment size in bits. `--segment-size 4831838208` is close to the segment limit of 4GB.

There is also a `--changed` flag, which prevents uploading of the object if the checksum has not changed:

```
swift --os-auth-type v3applicationcredential \
--os-application-credential-id <credential_id> \
--os-application-credential-secret <credential_secret> \
upload --changed --segment-size 4831838208 \
<container_name> <path>
```

Working with folders

There isn't "folders" when you managing the *Object Store* with the CLI. However, when you create an object, you may use the delimiter `/` to specify the path.

Accessing object store from KVM instances

KVM@TACC does not have its own Object Store. To use the *Object Store* from a KVM instance, you need to authenticate against a Chameleon site that provides one, such as CHI@TACC.

Use `ccauth` with `--all-sites` to generate credentials for all Chameleon sites at once:

```
ccauth clouds-yaml --all-sites --output ~/.config/openstack/clouds.yaml
```

This writes one cloud entry per site into your `clouds.yaml`. You can inspect the file to confirm which site entry to use.

Once you've identified the name of the site to use, export its cloud name and use standard OpenStack commands:

```
export OS_CLOUD=tacc
openstack container list
openstack container create <container_name>
openstack object create <container_name> <local_file>
openstack object save <container_name> <object_name>
```

20.2.3 Mounting object store as a file system

Important

The object store is no longer mounted automatically at boot. Before using `cc-mount-object-store` for the first time, authenticate with *ccauth* and run `setup-cc-mount-object-store` once to configure credentials. See *Changes to automatic credential setup on instances* for details.

Tip

rclone can upload small and large files to the object store, however, if you have trouble uploading larger objects, you may need to use the Swift CLI instead.

When logged into an instance using Chameleon-supported images, such as *CC-CentOS9-Stream <https://trovi.chameleoncloud.org/dashboard/artifacts/98de71d6-4c27-47d5-8a37-79e7847f90e8>_* and *CC-Ubuntu24.04 <https://trovi.chameleoncloud.org/dashboard/artifacts/b97e5f97-beeb-4109-9438-19c9c3886624>_*, you will find a README in the home directory for the *cc* user. The README describes how to mount containers in the Chameleon Object Store into a directory called `cc_my_mounting_point` in your home directory. Mounts are facilitated by the rclone tool. If the directory does not exist, this directory will be created the first time you mount a container. Inside the `cc_my_mounting_point` directory, you will find directories that map to containers you've mounted. If there is a directory inside `cc_my_mounting_point` that is not mounted it should have a file named `THIS_IS_NOT_MOUNTED` in it. Once you mount the container, the file will no longer be visible until the container is unmounted.

The tool can mount existing containers in the object store, or create them if they don't exist. The containers are from the specific site where the instance is located and only work at sites that have an object store (currently `CHI@UC` and `CHI@TACC`). For example, instances running at `CHI@UC` will interact with the object store also at `CHI@UC`. You will not be able to interact with object store data at other sites using this method.

Important

Some older Chameleon-supported images have an outdated mechanism for mounting the object store using `cc-cloudfuse`. This mechanism for mounting the object store is no longer recommended or supported. On older images you should use the Swift CLI directory to use the object store.

To mount, use the following command:

```
cc-mount-object-store start your_container_name
```

Now you can access your Chameleon Object Store as your local file system at: *~/cc_my_mounting_point/your_container_name*.

You can investigate if a mount is running for a container with:

```
cc-mount-object-store status your_container_name
```

You can also list all running mounts with:

```
cc-mount-object-store list
```

To unmount, use the following command:

```
cc-mount-object-store stop your_container_name
```

Important

Limitations

The primary usage scenario of the `rclone` tool is to allow you to interact with Chameleon Object Store using familiar file system operations. Because the tool runs on top of an object store, it is important to understand that not all functionality will behave identically to a regular file system.

1. Symbolic links, file permissions, and POSIX file locking operations are not supported.
2. Updating an existing file is an expensive operation as it downloads the entire file to local disk before it can modify the contents.
3. You can mount from multiple nodes, but there is no synchronization between nodes regarding writes to Object Storage.
4. The mounting root directory can only contain directories, as they are mapped to Object Store containers.
5. Renaming directories is not allowed.
6. It keeps an in-memory cache of the directory structure, so it may not be usable for large file systems. In addition, files added by other applications will not show up until the cache expires.

Keep these limitations in mind when considering the use of this tool to interact with the object store.

Warning

The use of `rclone` to sync files between your instance and the object store is a best effort tool. It is the responsibilty of the user to verify the files sync'd correctly and are valid.

Given the challenges of mapping files in a file system to an object store over a network, numerous problems can occur that may impact the availability of files on the object store. If you attempt to copy files into the mount point and receive errors, it is important that you verify the existence and contents of the file in the object store and not simply assume the file has been persisted there (even if it is present in the mount point).

CHAPTER

TWENTYONE

SHARES

Chameleon provides a shared file system service through the OpenStack Manila interface. With the service, you can create a shared file system, mount to the bare metal instances, and manage some of its properties, such as visibility.

Hint

Chameleon shared file system service is currently backed by a CephFS. Same as our *object store* service, the data is replicated and the replication should provide good availability in case of hardware failures.

Difference between shared file system and object store

You can choose either shared file system or object store to store, manage, and share your data with your collaborators. The object store is suitable for storing large objects at scale, but isn't suitable for transactional data, as objects are immutable and updated in their entirety. Chameleon shared file system instead provides a NFS mount to the bare metal instances, with the NFS protocol managing locking and data integrity processes required to provide multiple concurrent access to data.

The shared file system service is available at CHI@UC and CHI@TACC. Each region has its own service and the shares created at one region are not available to the other. As all other Chameleon services, you can create and manage your shares using both GUI and CLI.

Note

Shares can also be created and managed programmatically via the chi.storage module in python-chi — see our *Jupyter and python-chi guide* for an introduction.

21.1 Shares concepts

21.1.1 Storage networks

To provide isolation among shares created by different projects, accessing a share requires a storage network, which are special networks you can reserve to use. When reserving a storage network, add *usage_type=storage* to the resource properties. To learn more about reserving networks, read the *reservations documentation*. All bare metal instances that are created on the storage network have access to all the project shares.

Tip

To attach floating IP to your instance created on a storage network, you need to create a router with *public* external network. Then connect the storage subnet to the router. You must specify an unused IP address which belongs to the selected subnet. To learn more about creating router and connecting subnet, read *isolated network VLANs*.

21.1.2 Shares

21.1.3 Visibility

Shares are owned by the project. By default, all shares have *private* visibility and can only be listed and accessed within your project. All bare metal instances owned by the project have read and write permissions to the project's shares. You can also make your shares *public*. All Chameleon users and projects can list public shares, and with a storage network, all projects have read-only access to a public share.

21.1.4 Accessibility

A share is a pre-allocated storage space at a CephFS. You can *mount your shares to your bare metal instances via NFS protocol*. The accessibility of the shares are controlled internally by the reservation service. You are not allowed to edit the access rules of a share.

21.1.5 Quotas

We do not charge SUs for the storage spaces of your shares. However, we do limit the total size and the number of shares you can create within your project. The maximum number of shares is 10 and the maximum size allowed for all shares in a project is 2000 GiB. If you need to increase the default quota, submit a ticket via the Help Desk.

21.2 Managing shares using GUI

To manage your share, use the *Shares* page at CHI@UC or CHI@TACC by navigating to *Project* > *Share* > *Shares*.

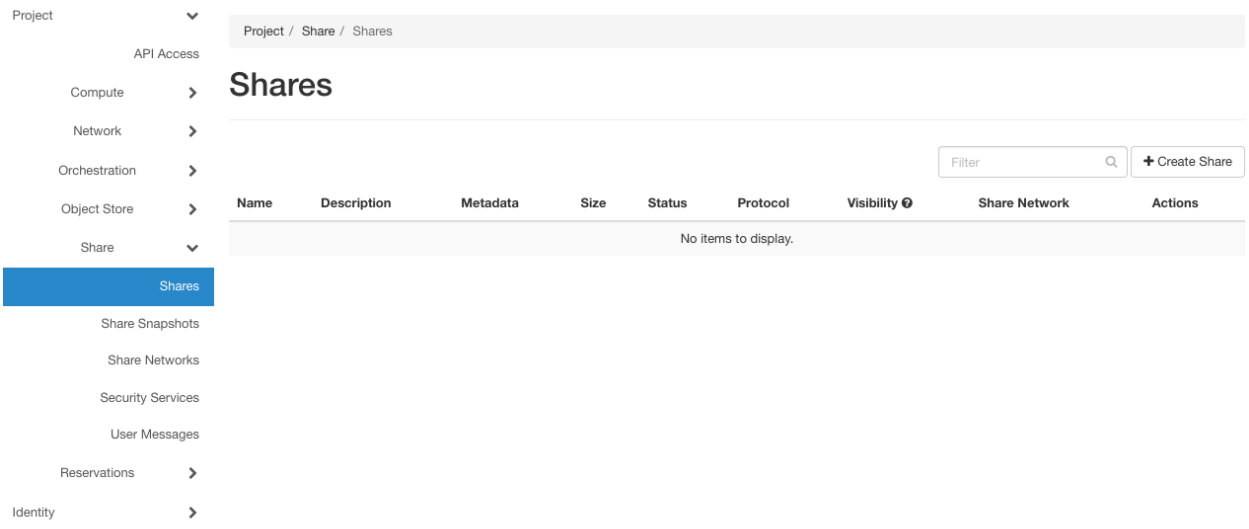


Fig. 1: The Shares page

21.2.1 Create share

Click the *Create Share* button. In the *Create Share* dialog, provide a name and the size of your share, and then click the *Create* button to create a share.

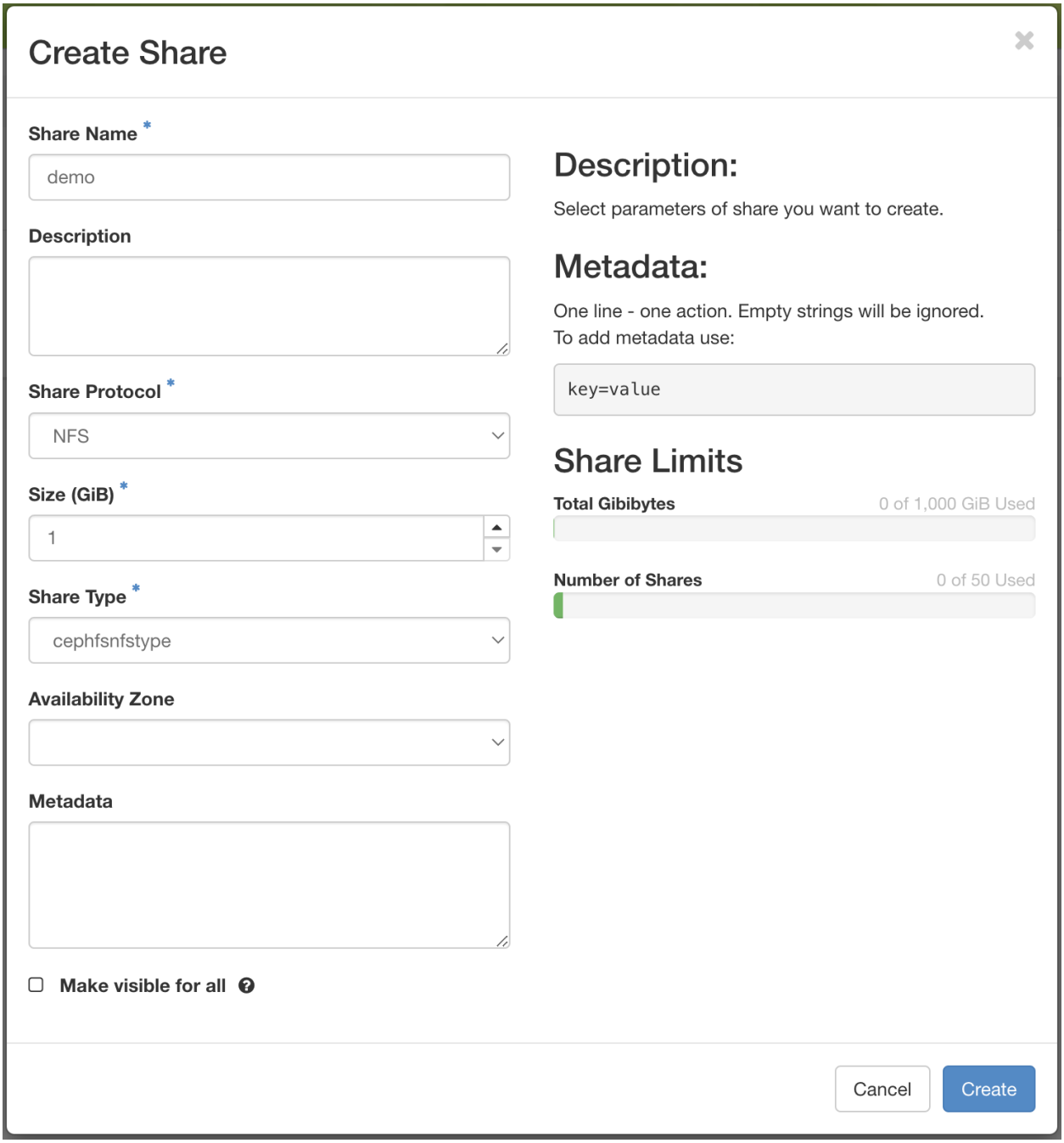


Fig. 2: The Create Share dialog

> **Note**
>
> A storage network is not required for creating shares. It's only required to access the shares.

21.2.2 View share

You can look at the details of a share by clicking the share name in the *Shares* page. Note that the paths of the *export locations* are important as you will use this path to mount your share to your bare metal instances. You can also see the other properties, such as visibility and size. The access rules are listed in the *share details* page, though you can not edit the rules, as they are controlled by the reservation service.

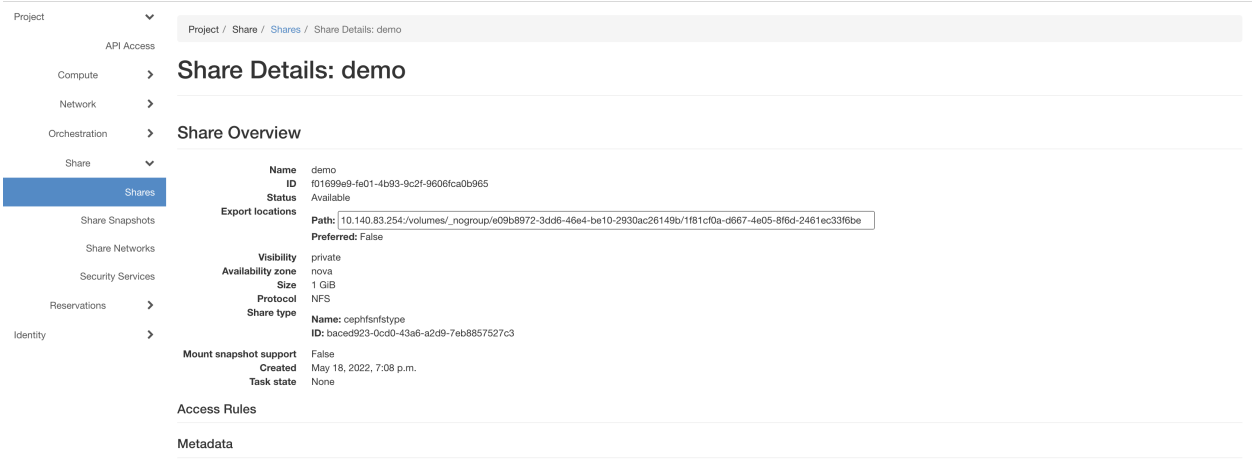


Fig. 3: The Share details

21.2.3 Edit share

You can manage the properties and extend the size of a share by clicking the *Action* dropdown in the *Shares* page.

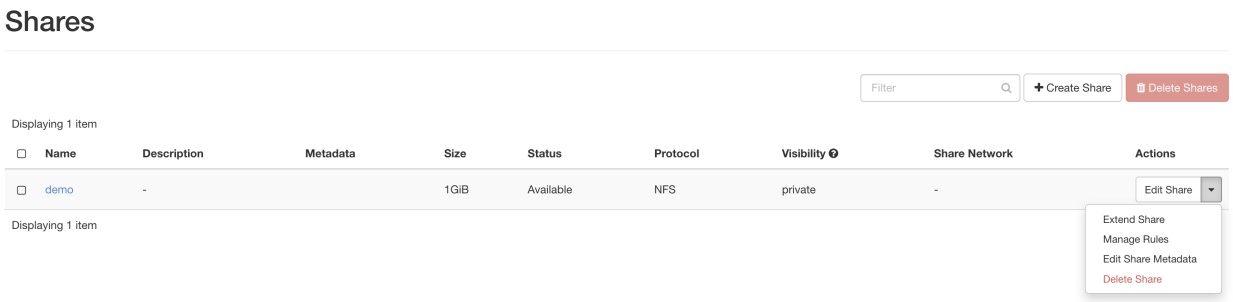


Fig. 4: The Action dropdown

21.2.4 Delete share

You can use the *Action* dropdown to delete a single share, or select multiple shares and click the *Delete Shares* button.

Important

Be careful when deleting shares, as the action is irreversible. However, the termination of your storage network reservation **DOES NOT** delete your share. Your shares persist until you manually delete them.

21.3 Managing shares using CLI

As all other Chameleon services, you can manage your shares via CLI as well.

Tip

Reading *Command Line Interface (CLI)* is highly recommended before continuing on the following sections.

In addition to installing the CLI, you must also install *python-manilaclient* package:

```
pip install python-manilaclient
```

Then, you must set environment variables for your account and project using *The OpenStack RC Script*.

Tip

If you get HTTP 406 error of `version is not supported by the API`, add `--os-share-api-version 2.65` to the command to specify manila minor version.

21.3.1 List shares

To list all shares of your project, run the following command:

```
openstack share list
```

You can filter the results by the share name via adding a `--name` argument to the list command.

21.3.2 Create share

To create a share, using the following command:

```
openstack share create --name <name of your share> NFS <size in GiB>
```

For example, for creating a 1 GiB share with name of `my-first-share`, run:

```
openstack share create --name my-first-share NFS 1
```

Note

Only the NFS protocol is supported.

You can add the `--public true` to make your share public.

21.3.3 Edit share

To change the visibility of a share, run:

```
openstack share set --public <true/false> <name/id of the share>
```

To update the name or the description of a share, run:

```
openstack share set --name <new name> --description <description> <name/id of the share>
```

To extend/shrink the size of a share, run:

```
openstack share resize <name/id of the share> <new size in GiB>
```

21.3.4 View share

To view the details of a share, run:

```
openstack share show <name/id of the share>
```

21.3.5 Delete share

To delete a share, run the following command:

```
openstack share delete <name/id of the share>
```

21.4 Mounting shares to instances

In order to allow your instances to access the share, you need to create your instances using the *pre-reserved storage network*. To learn more about how to create a bare metal instance on a network, read *the bare metal instances section*.

Important

The shares are independent of the storage networks. You can create shares any time regardless of the status of the storage networks. The storage networks are only used to access your data stored in the share.

After your instance becomes active, find the export location path of the share using *GUI* or *CLI*. To mount the share, run the following command:

```
sudo mount -t nfs -o nfsvers=4.2,proto=tcp <export location path> <mount dir>
```

Now, you can read and write to the share and it behaves identically to a regular file system.

To unmount, run the following command:

```
sudo umount <mount dir>
```

CHAPTER

TWENTYTWO

NETWORKING

Networking on Chameleon is implemented using OpenStack Neutron. Most experiments will require *Basic networking* functionality including Internet access and connectivity between nodes. Chameleon provides basic networking capabilities via a pre-configured shared network called `sharednet1`. Many experiments require additional connectivity and control of the network. These experiments can utilize Chameleon's advanced networking capabilities including *Isolated network VLANs*, *External layer 3 connectivity*, and *External layer2 connections (stitching)*.

Not sure which advanced option you need? A quick way to decide:

- Want your own private layer-2 network among your Chameleon nodes, with full control over addressing, DHCP, and routing — for example, to run your own DHCP server or isolate a multi-node experiment from other users' traffic? Use *Isolated network VLANs*.
- Want your Chameleon nodes to talk to nodes at another Chameleon site (or to FABNET) at layer 3, without managing floating IPs or a stitched VLAN yourself? Use *External layer 3 connectivity*.
- Want a dedicated layer-2 connection between a Chameleon network and an external facility like FABRIC, so you can run your own layer-3 (or lower) protocols over it? Use *External layer2 connections (stitching)*.

Note

Networks can also be created and managed programmatically via the chi.network module in python-chi — see our *Jupyter and python-chi guide* for an introduction.

22.1 Basic networking

Note

Step-by-step instructions for getting started with Chameleon are available in the getting-started section of this documentation. These instructions include using basic networking functionality.

22.1.1 Can't reach your instance?

If you can't connect to an instance over SSH or another port, there are **multiple independent layers** that can each block the connection on their own — fixing one does not mean the others are open. Check them in order:

1. **Is a Floating IP associated with the instance, on a network with a router?** See *Floating IP addresses* below.
2. **(KVM@TACC only) Is a Security Group applied that allows the port?** Security Groups only exist on KVM@TACC — bare metal instances at CHI@TACC and CHI@UC have no Security Group menu at all, so this step doesn't apply to them. See *Security groups* below.

3. **Is the port open in the instance's own firewall?** Every Chameleon-supported image runs a firewall *inside* the instance (`firewalld`, or `ufw` on some older images), independently of any Security Group, and it only allows SSH (port 22) by default. This is the step people miss most often: opening a Security Group does **not** open the instance's internal firewall, and opening the internal firewall does **not** open a Security Group — on KVM@TACC you need both. See *Firewall* below.
4. **Are you using the right key pair?** `Permission denied (publickey)` usually means the instance was launched with a different key than the one you're connecting with.

22.1.2 Shared network

All Chameleon Projects have access to the fixed network `sharednet1` which is used by most experiments. The `sharednet1` is a pre-configured network shared among all Chameleon Projects with one *Subnet* and includes a *Router* providing NAT access to the public Internet. All instances using `sharednet1` can communicate directly.

Tip

`sharednet1` is a shared, finite pool of addresses. If you're unable to launch an instance on `sharednet1` because it has no free addresses, you don't have to wait — create your own *isolated VLAN network* instead. It gets you a dedicated subnet under your own control, with no dependency on `sharednet1` capacity.

22.1.3 Multiple networks

Some Chameleon bare metal nodes support connecting to multiple networks. Currently, the number of networks allowed is limited to the number of enabled NICs on the node (currently this is up to 2). It is possible to find such nodes via *Resource Discovery* by filtering by the "Enabled" flag for a given Network Adapter slot. Note that the slots are 0-indexed, meaning the first NIC is referred to as Network Adapter #0.

When launching a node that supports multiple networks, simply assign multiple networks to the instance when you are launching it. The networks will be mounted on NICs in the same order that the networks are assigned; that is, the first assigned network will be mounted on Network Adapter #0, and the second on Network Adapter #1, and so on.

Tip

Worked example: keep SSH access while using a second NIC for your experiment. A common pattern is to put `sharednet1` (or any network with a router and floating IP access) on Network Adapter #0 so you retain normal SSH/management access, and put a dedicated *isolated network* on Network Adapter #1 for your experiment's data path. To do this via the CLI, list both networks in the order you want them assigned when creating the server:

```
openstack server create \
  --nic net-id=<sharednet1_id> \
  --nic net-id=<experiment_network_id> \
  --image <image> \
  --flavor baremetal \
  --key-name <key_name> \
  --hint reservation=<reservation_id> \
  my-instance
```

`sharednet1` will come up as the first interface (e.g. `eno1`) and your isolated network as the second (e.g. `eno2`). Keep managing the instance over the `sharednet1` interface, and configure your experiment traffic on the second one.

22.1.4 Floating IP addresses

Instances on Chameleon are assigned a *fixed* IP address that can be used for local connectivity as well as NAT access to the public Internet. A publicly accessible IPv4 address (*Floating IP address*) is required in order to access Chameleon instances from the Internet or host public services. CHI@TACC and CHI@UC each have a limited number of public IP addresses that can be allocated to your instances.

The getting-started guide shows how to allocate *Floating IP address* to your nodes.

Important

The Chameleon floating IP address pool is a shared and finite resource. **Be responsible and release any unused floating IP address, so other Chameleon users and projects can use them!**

Note that in order to use a *Floating IP address*, your instance must be connected to a network with a router providing external connectivity. The default network `sharednet1` is one such network, but it will also work with isolated networks that you create. For more information about VLAN and isolated networks, see *Isolated network VLANs*. When associating a floating IP, if you do not see your instance in the dropdown, double-check that it is connected to an appropriate network.

In addition to the ad-hoc method described above, Floating IP addresses can also be reserved via the Reservation *GUI* or *CLI*. These ad-hoc and reserved addresses are drawn from **separate pools**.

Tip

If you get an error that no Floating IP addresses are available, this usually means one pool is exhausted, not both. Try the other method — if ad-hoc allocation fails, try reserving one instead (or vice versa).

Releasing floating IP addresses via GUI

To release floating IP addresses through the GUI:

1. Navigate to *Network* > *Floating IPs* in the sidebar
2. To release a single IP: click the dropdown in the *Actions* column and select *Release Floating IP*
3. To release multiple IPs: select them via checkboxes and click the *Release Floating IPs* button

You can also release floating IP addresses via the command line using `openstack floating ip delete <floating-ip>`.

Floating DNS records

Each Floating IP is also contained within a dedicated DNS A record; this means that you can access your instance over the Internet either via its Floating IP or a special hostname. This can be particularly helpful if you want to set up a TLS certificate for HTTPS to secure a service you are exposing over the web, e.g., with LetsEncrypt.

Site	Hostname pattern (for IP *AA.BB.CC.DD*)
CHI@TACC	**chi-dyn-AA-BB-CC-DD.tacc.chameleoncloud.org** *e.g., chi-dyn-129-114-108-147.tacc.chameleoncloud.org*
CHI@UC	**chi-dyn-AA-BB-CC-DD.uc.chameleoncloud.org** *e.g., chi-dyn-192-5-87-98.uc.chameleoncloud.org*

22.1.5 Security

When your instance has a *Floating IP address* assigned, it is reachable directly over the public Internet. For this reason, it is important to consider the security of any services running on your instance. In particular, **ensure that you have not allowed SSH authentication with passwords** (this is disabled by default on Chameleon-supported images.)

In order to better protect Chameleon instances, Chameleon Ubuntu and CentOS base images come with baked-in firewall rules which severely restrict connections over the public internet. If using a different image or if you disable firewall rules, realize that any network services can potentially be exposed to the public Internet if your instance has a Floating IP attached.

Warning

Some commodity systems such as Apache Spark and Hadoop have in the past shipped with *very insecure* default settings. Pay particular attention to the security needs of your experiment when selecting what systems you need to install on your node, particularly when exposing the node to the Internet.

Note

We're here to help! If you want advice on how to securely run your experiment, feel free to file a Help Desk ticket.

Firewall

Chameleon-supported Ubuntu and CentOS images are preconfigured with a firewall utility called `firewalld` enabled.

Note

For Chameleon supported images (e.g., CC-Ubuntu24.04) built in April 2025 or later, the firewall is disabled on instances deployed on KVM since those instances use security groups for the firewall. `firewalld` is enabled on all bare metal instances in all other regions. Images built before April 2025 all have `firewalld` enabled by default.

It has the following rules set:

```
# sudo firewall-cmd --zone=public --list-services
dhcpv6-client ssh
```

These rules allow ssh traffic on port 22 over the public internet.

Warning

By default, all firewall changes are **temporary**, and will be lost on instance reboot. This is a safety mechanism to avoid locking yourself out. To make changes **permanent**, execute:

```
sudo firewall-cmd --runtime-to-permanent
sudo firewall-cmd --reload
```

To enable HTTP/HTTPS on port 80 and 443:

```
sudo firewall-cmd --zone=public --add-service http
sudo firewall-cmd --zone=public --add-service https
```

Firewalld has many "built-in" rules for common services, but you can also enable communication over a specifc port using the command:

```
# list all open ports
sudo firewall-cmd --zone=public --list-ports

# open a new port
sudo firewall-cmd --zone=public --add-port=<port>/<protocol>

# example
sudo firewall-cmd --zone=public --add-port=9001/tcp
```

You can also permit connections from a specific ip or network, such as a trusted endpoint, or within your own isolated networks on Chameleon.

```
sudo firewall-cmd --zone=trusted --add-source=<your_subnet_cidr/netmask>
```

To enable this by default for all private IP ranges, you can do the following, but note that this can be insecure on shared or routed networks (sharednet1, sharedwan1 and similar).

```
sudo firewall-cmd --zone=trusted --add-source=192.168.0.0/16
sudo firewall-cmd --zone=trusted --add-source=172.16.0.0/12
sudo firewall-cmd --zone=trusted --add-source=10.0.0.0/8
```

Any other incoming connections will be denied.

For more examples and information, see:

- Ubuntu's man page for firewalld
- Fedora Linux Guide
- Rocky Linux Guide

Security groups

> **Attention**
>
> Security Groups are a **KVM@TACC-only** feature. Bare metal instances at CHI@TACC and CHI@UC do not have a Security Group menu at all — on those sites, the in-instance *Firewall* described above is your only network-level access control.

KVM@TACC supports *Security Groups*, which can be assigned directly to instances upon launch or after the instance is already running. Every instance is assigned the `default` Security Group unless you specify otherwise, and **that group blocks all inbound traffic, including SSH, by default** — it only allows outbound traffic. To reach an instance over SSH, you must apply a Security Group (e.g. the built-in "Allow SSH" group) that explicitly permits inbound TCP port 22. See *Security groups* for how to create and apply one via the GUI, or *Work with KVM using the CLI* for the CLI equivalent.

Limit bound interfaces

Instead of binding a web service to all interfaces (e.g. `0.0.0.0` for IPv4, `::` for IPv6), consider listening only on the node's private IP, which is not routable from the public Internet. If you can, listening on localhost (`127.0.0.1`) is even safer. Most web services have a way to specify the bind address and some default to binding on all interfaces, which is often insecure.

22.2 Isolated network VLANs

By default, bare metal nodes on each Chameleon site share the same local network (shared VLAN and IP subnet). However, some experiments may require more network isolation, which is now supported by Chameleon.

Chameleon's implementation of network isolation is based on dynamically managed VLANs (network layer 2) associated with user-configured private IP subnets (network layer 3). This means that all network communications local to the IP subnet or the broadcast domain (such as Ethernet broadcast, ARP, IP broadcast, DHCP, etc.) will be restricted to the user-configured network and its associated VLAN. This feature enables a range of experiments in networking and security. For example, this allows running your own DHCP server to configure virtual machines running on bare metal nodes, without impacting other users.

Note

- Strong network isolation is provided at network layer 2 only. Even using separate IP subnetworks, any bare metal node can still communicate with each other and with the Internet through the network's router. We are investigating solutions to provide stronger isolation at network layer 3.
- Network isolation works on all nodes, including our low-power HP Moonshot nodes (low-power Xeon, Atom, ARM64).

To use this feature, you will need to create a dedicated network and router. You can use a *Heat template*, use the *Network* panel of the GUI, or use the CLI.

22.2.1 Configuring networking using a Heat template

1. Go to *Project* > *Orchestration* > *Stacks*.
2. Click the *Launch Stack* button to open an interactive dialog.
3. Select *URL* as *Template Source* and paste https://raw.githubusercontent.com/ChameleonCloud/heat-templates/master/network-isolation/network-isolation.yaml to *Template URL*.
4. Click the *Next* button to navigate to the *Launch Stack* dialog.
5. Provide a name for your stack, enter your password, and set a private IP range, such as 192.168.1.0/24.
6. Set the first and the last IP addresses of *DHCP* range.

   Important

   The first IP adddress in the DHCP range should never be *.1 and *.2. The last IP address in the range must be less than *.255.

7. Start creating the network and router by clicking the *Launch* button.

22.2.2 Creating a network using the GUI

To create a Network from either the *Network Topology* page or the *Networks* page, click the +*Create Network* button to open the *Create Network* dialog.

In *Create Network* dialog, name your network. In general, you will also want to create a *Subnet* for your new Network, so make sure you have *Create Subnet* checked. Click the *Next* button.

When creating a *Subnet*, you must specify a *Subnet Name* and a CIDR *Network Address* that contains a private IP address and a subnet mask length. For example, you may create a Class C subnet with a 24-bit mask by entering `192.168.1.0/24`. You may set a Gateway or leave it blank to use the default. Then, click the *Next* button.

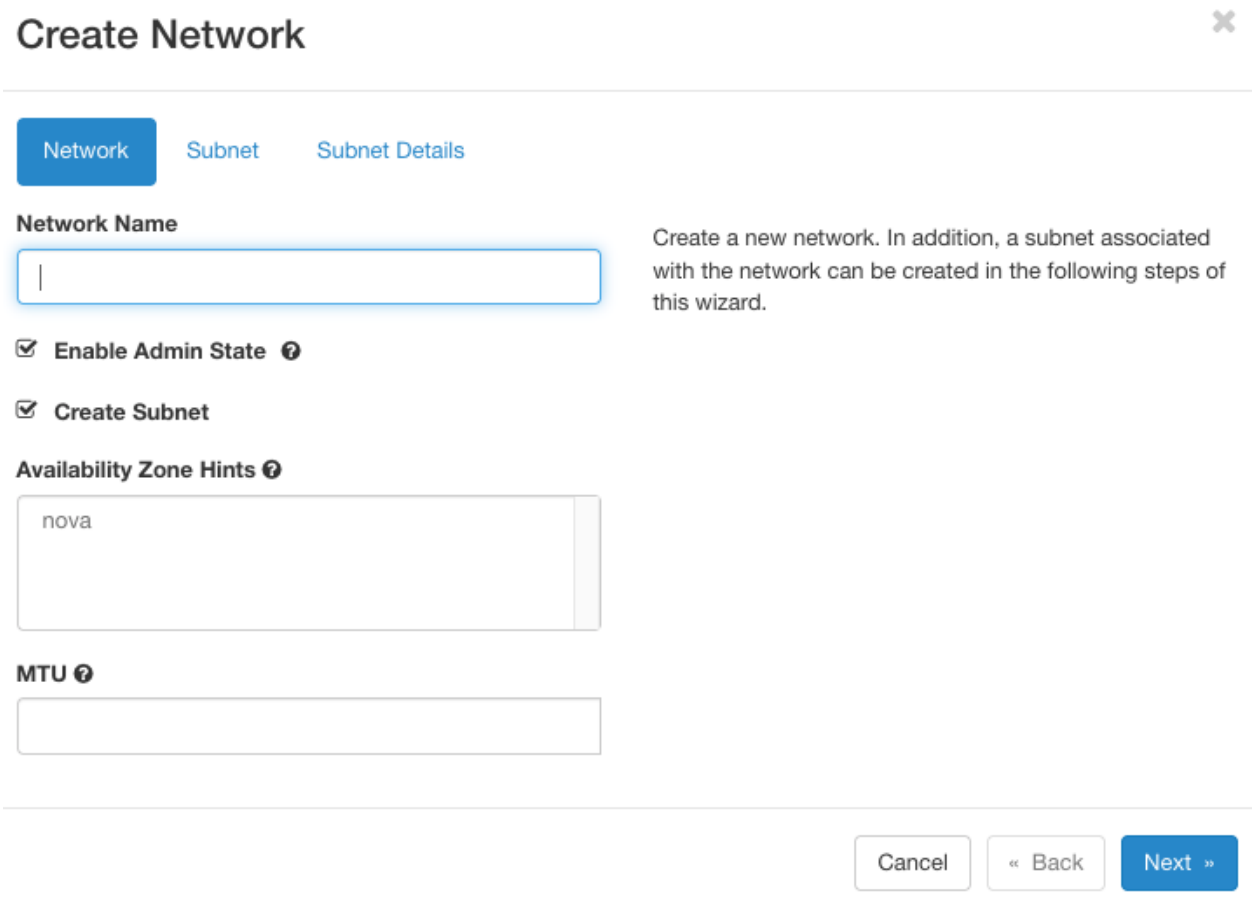


Fig. 1: The Create Network dialog

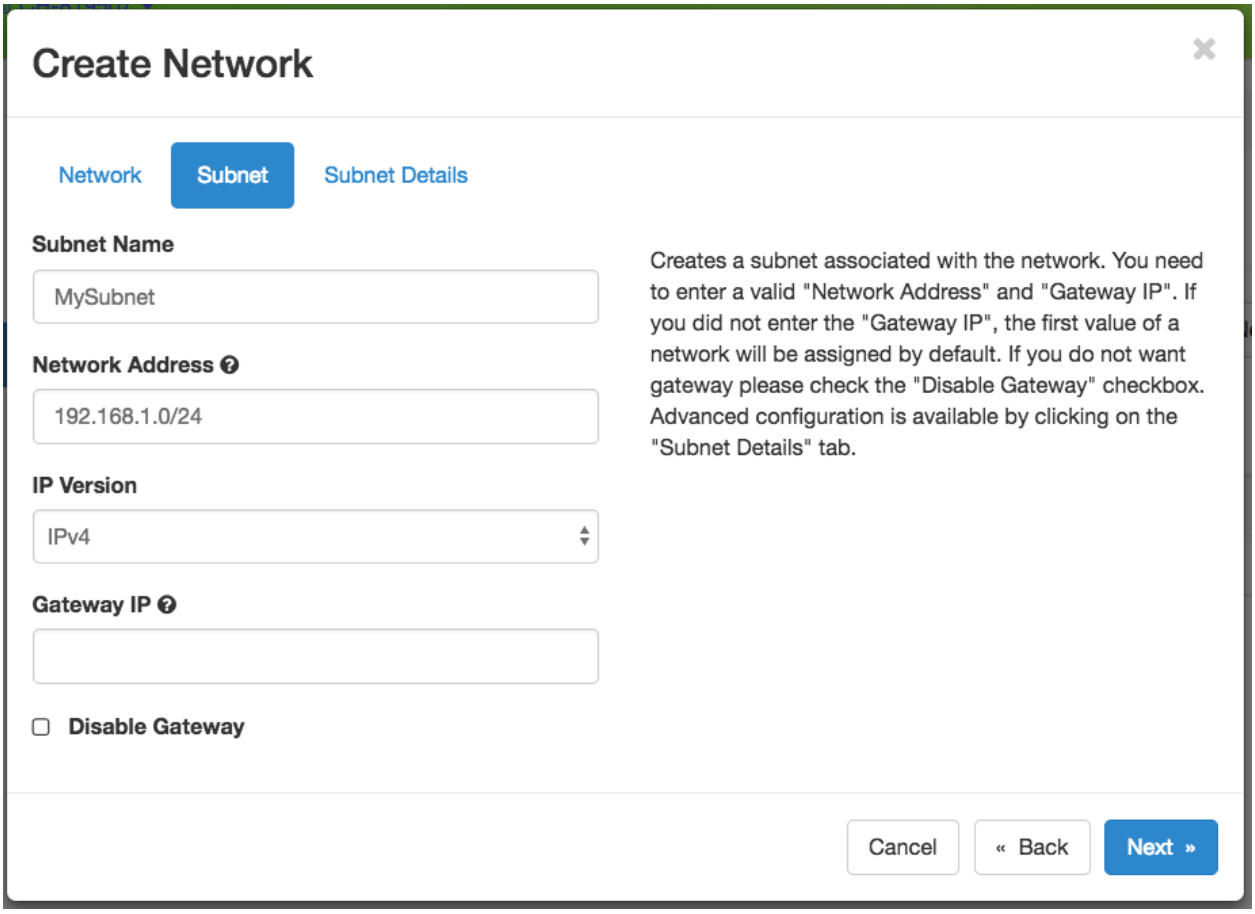


Fig. 2: The Subnet tab

Attention

Do not select the *Disable Gateway* checkbox!

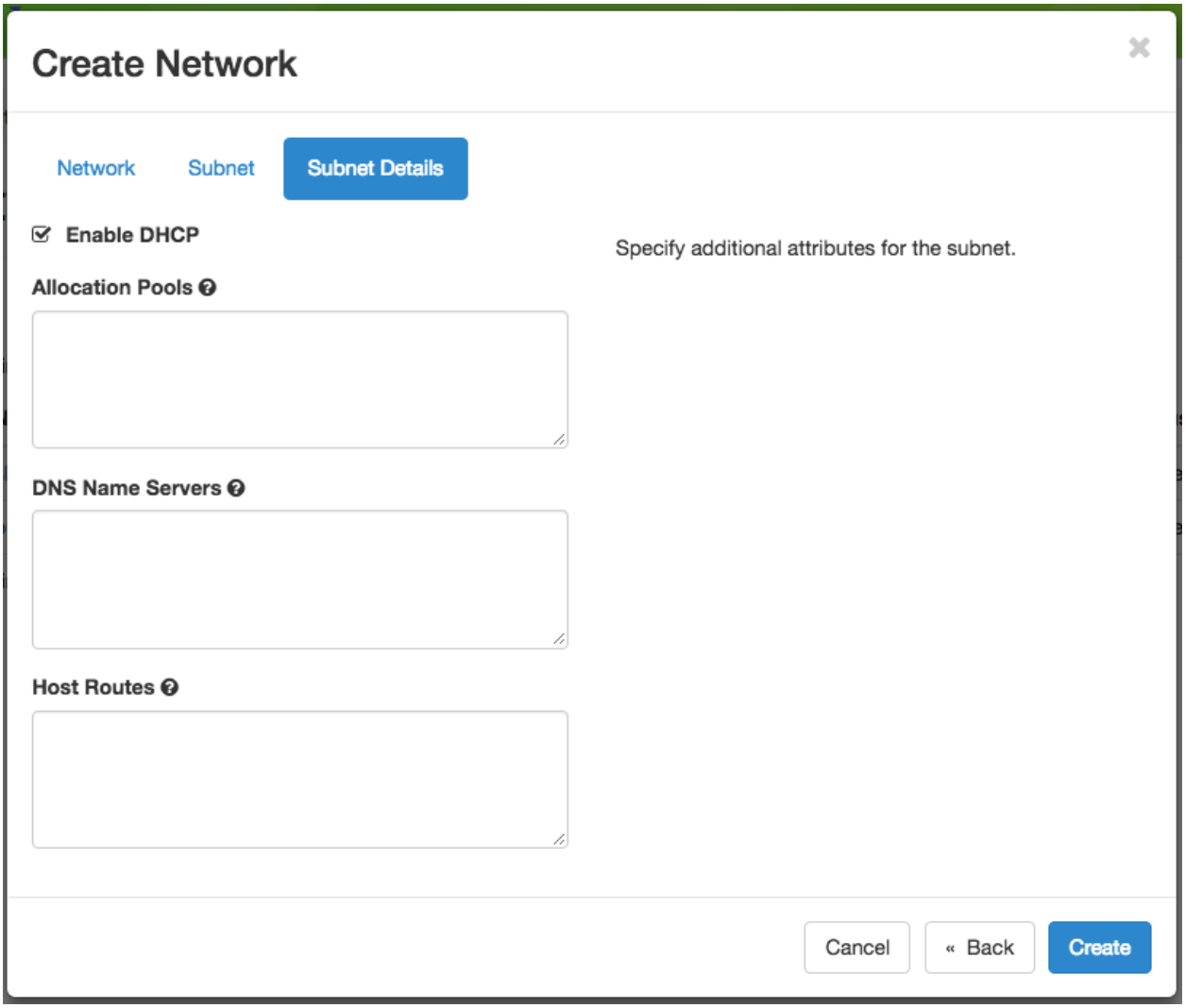


Fig. 3: Subnet details

Tip

If you plan to run your own DHCP server on this network, either skip creating a subnet, or uncheck *Enable DHCP* in the *Subnet Details* below. Neutron's built-in DHCP will conflict with your own DHCP server if both are active on the same network. This can also be handy if you're setting static IPs, and just want to avoid noise in the logs when running e.g. tcpdump.

You may specify *DHCP* and static *Route* information at *Subnet Details* section:

- *Allocation Pools* section allows you to specify *DHCP* address ranges in the format of `<first address>,<last address>`. For example, entering `192.168.1.2,192.168.1.100` will create a *Subnet* with IP ranges from

`192.168.1.2` to `192.168.1.100`.

- *DNS Name Servers* section allows you to specify a list of DNS servers.

> **Note**
>
> At CHI@TACC, use `129.114.97.1` and `129.114.97.2` for your DNS servers At CHI@UC, use `8.8.8.8` and `8.8.4.4` for your DNS servers

- *Host Routes* section allows you to specify static routing information for the subnet in the format of `<subnet CIDR>,<router IP address>`. For example, `192.168.3.0/24,10.56.1.254` means all traffic from this Subnet to `192.168.3.0` will be forwarded to the Router Interface at `10.56.1.254`.

> **Note**
>
> All three sections above are line separated.

Click *Create* button and a new Network will be created. Check if the network is created without error.

Creating a router

To create a *Router* from either the *Network Topology* page or the *Routers* page, click the *+Create Router* button to open the *Create Router* dialog.

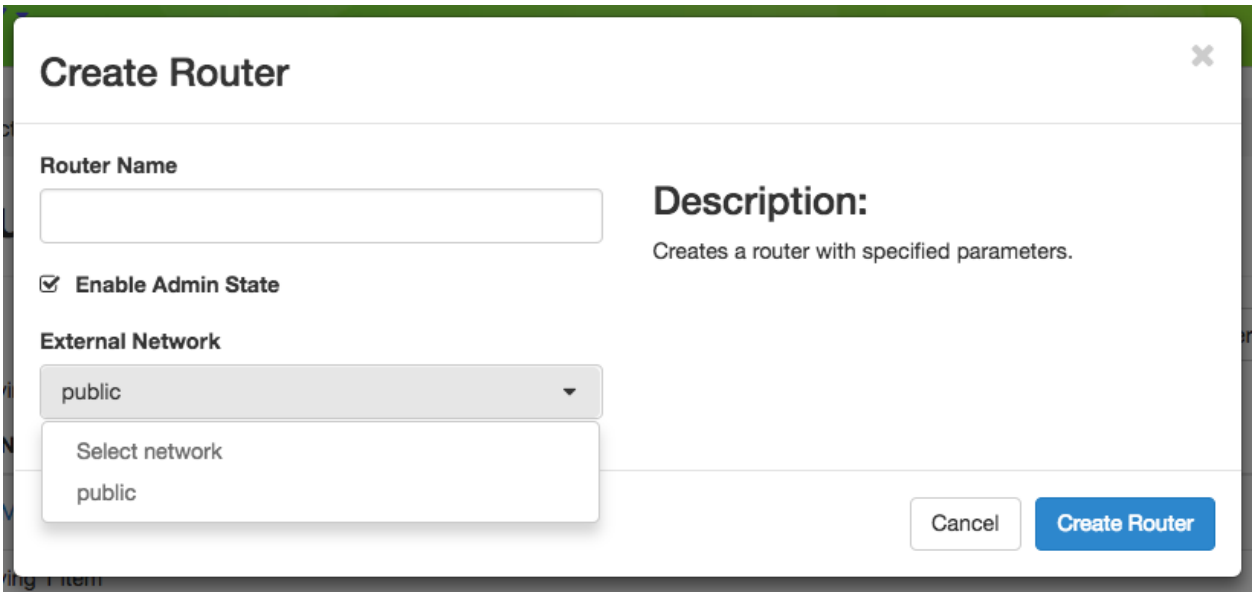


Fig. 4: The Create Router dialog

In this dialog, specify a *Router Name*. Optionally, you may select *public* as the *External Network* if you want to have external access. Click *Create Router* to complete the process.

Adding a router interface

A Router may have multiple *Interfaces*, each connected to a *Network*. You may add an *Interface* to an existing *Router* by clicking on *Add Interface* from either the *Network Topology* page or the *Routers* page to open the *Add Interface* dialog.

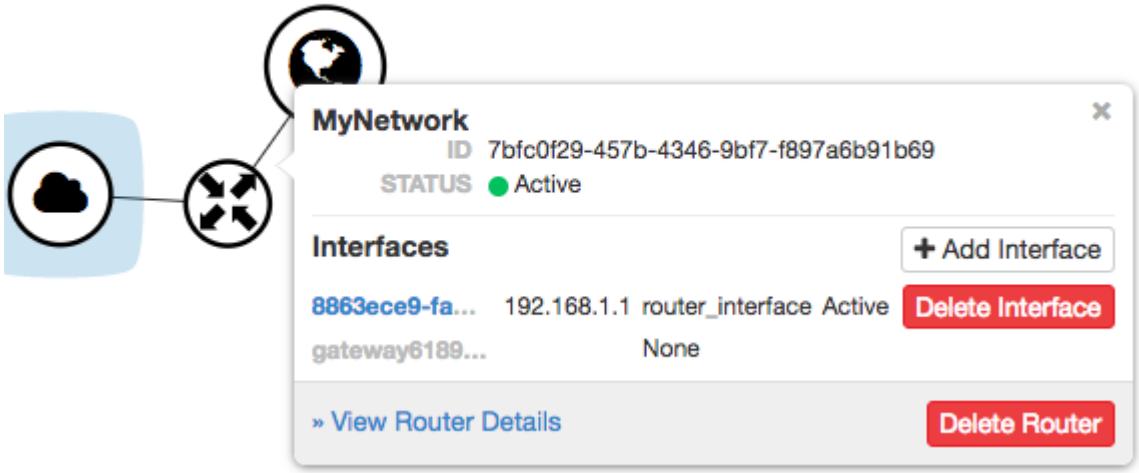


Fig. 5: The Router interface in the Network Topology page

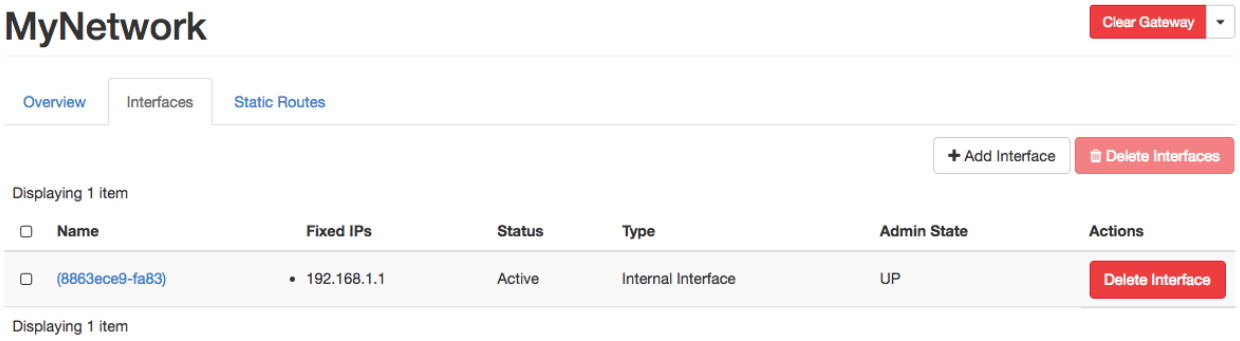


Fig. 6: The Interfaces tab in the Router detail page

First, select a network and subnet you have created. You can specify an *IP address*; otherwise, Chameleon will attempt to assign an IP address automatically. The gateway IP you assigned to the subnet will be automatically picked.

Deleting networking objects

Attention

Network objects such as *Routers* and *Networks* must be deleted in the reverse order of which they were created. Objects **can not** be deleted while other objects are depending on them.

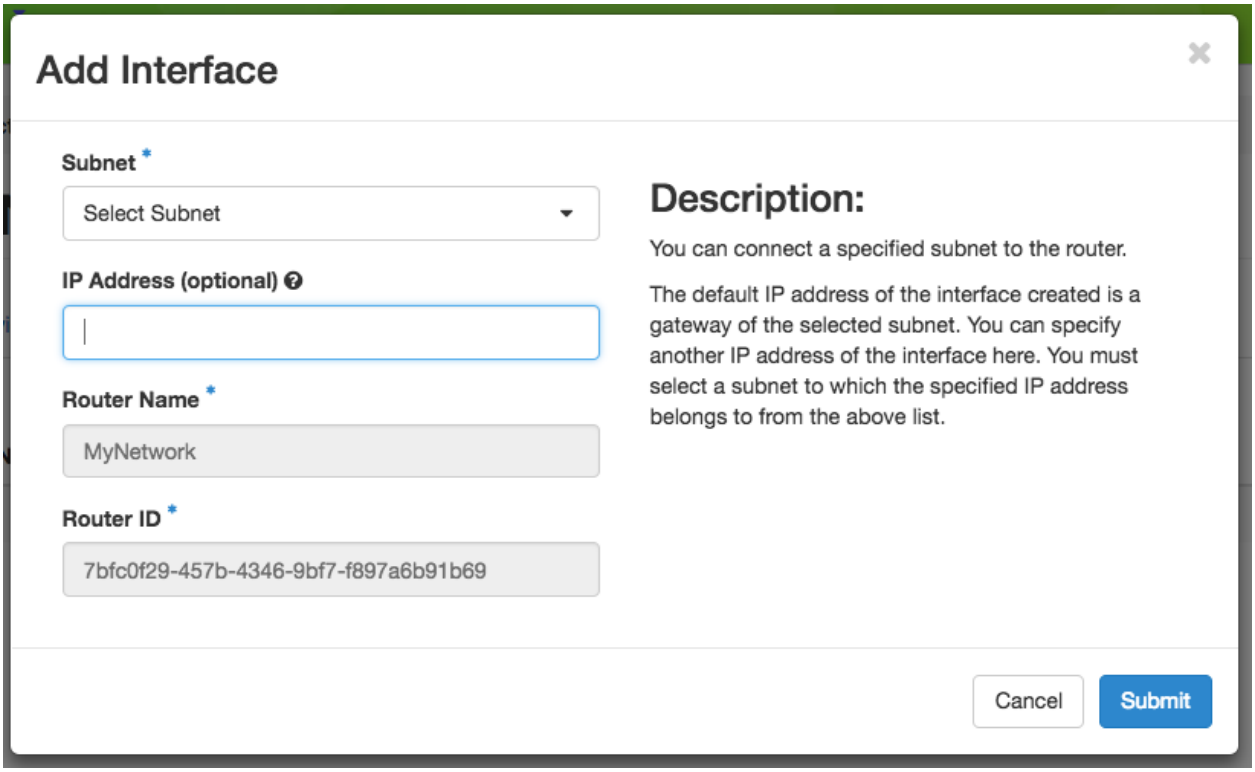


Fig. 7: The Add Interface dialog

Attention

Before starting to delete network objects, make sure all instances using them are terminated!

1. Go to *Project* > *Network* > *Routers*, and click on the router you would like to delete.
2. Go to *Static Routes* tab, and click on the *Delete Static Routes* button in the *Action* column. The *Static Routes* will be deleted after confirm.
3. Go to *Instances* tab, delete the Gateway interface by clicking on *Delete Interface* button in the *Action* column and confirm the deletion.
4. Now you can safely delete the router by clicking on the dropdown on the upper right corner. Then, click on *Delete Router*. Finally, confirm your deletion of the router.
5. Go to *Project* > *Network* > *Networks*, and delete the network by using the dropdown in the *Action* column. Alternatively, you may delete the network by selecting the network using the checkbox and click on *Delete Networks* button on the upper right corner. Confirm your deletion to finish the process.

22.2.3 Configuring networking using the CLI

Tip

Reading *Command Line Interface (CLI)* is highly recommended before continuing on the following sections.

Before using the CLI, make sure you have configured environment variables using *The OpenStack RC script*.

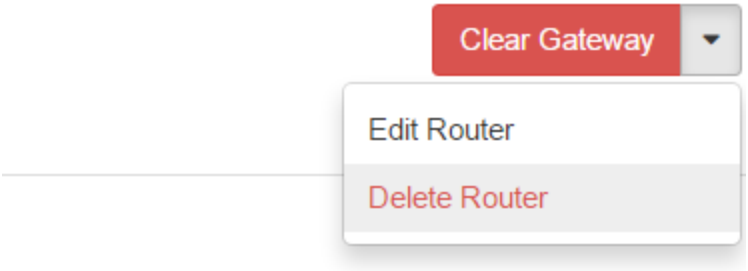


Fig. 8: Dropdown for deleting router

Note

There are two different ways to get an isolated VLAN network, and they're not interchangeable:

- **Create it directly** (below) — Neutron picks any available network segment for you automatically. This is the simplest path for most experiments.
- **Reserve a specific segment via :ref:`reservation-cli-vlan`** — use this instead if you need to guarantee a *specific* VLAN segment (for example, to match one already configured on external hardware), or if direct creation fails because Chameleon has no free segments left at the moment (see the note below). Either way, once the network exists you still configure its subnet and router the same way, as described in this page.

Creating a network

You can create an *Isolated* VLAN Network using the command:

```
openstack network create --provider-network-type vlan --provider-physical-network␣
↪physnet1 <network_name>
```

The output should look like the following:

```
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | UP                                   |
| availability_zone_hints   |                                      |
| availability_zones        |                                      |
| created_at                | 2018-03-23T23:45:19Z                 |
| description               |                                      |
| dns_domain                | None                                 |
| id                        | 21ed933c-323d-4708-930c-d5f82c507430 |
| ipv4_address_scope        | None                                 |
| ipv6_address_scope        | None                                 |
| is_default                | None                                 |
| is_vlan_transparent       | None                                 |
| mtu                       | 1500                                 |
| name                      | MyNetwork                            |
| port_security_enabled     | False                                |
| project_id                | d5233415ee0b467baec14cbd2d0e1331     |
```

(continues on next page)

(continued from previous page)

```
| provider:network_type      | vlan                                 |
| provider:physical_network  | physnet1                             |
| provider:segmentation_id   | 2018                                 |
| qos_policy_id              | None                                 |
| revision_number            | 2                                    |
| router:external            | Internal                             |
| segments                   | None                                 |
| shared                     | False                                |
| status                     | ACTIVE                               |
| subnets                    |                                      |
| tags                       |                                      |
| updated_at                 | 2018-03-23T23:45:19Z                 |
+----------------------------+--------------------------------------+
```

Note

Note the `provider:segmentation_id` field in the above output. Each *Isolated* VLAN Network requires a unique network segment to operate. There are a finite number of valid network segments on Chameleon. If you are unable to create a network because there are no valid network segments available, then you can create a network automatically by *Creating a lease to reserve a VLAN segment*.

Once you have created a Network, you may create a subnet with the command:

Tip

If you plan to run your own DHCP server on this network, either skip creating a subnet, or use `--no-dhcp` instead of `--dhcp` below. Neutron's built-in DHCP will conflict with your own DHCP server if both are active on the same network segment.

```
openstack subnet create --subnet-range <cidr> --dhcp --network <network_name> <subnet_
↪name>
```

For example, the command:

```
openstack subnet create --subnet-range 192.168.1.0/24 --dhcp --network MyNetwork MySubnet
```

will create a subnet with the following output:

```
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| allocation_pools  | 192.168.1.2-192.168.1.254            |
| cidr              | 192.168.1.0/24                       |
| created_at        | 2018-03-23T23:50:11Z                 |
| description       |                                      |
| dns_nameservers   |                                      |
| enable_dhcp       | True                                 |
| gateway_ip        | 192.168.1.1                          |
| host_routes       |                                      |
| id                | 8be4e80d-ba49-4cdc-8480-ba43dd4724c2 |
```

(continues on next page)

(continued from previous page)

```
| ip_version        | 4                                    |
| ipv6_address_mode | None                                 |
| ipv6_ra_mode      | None                                 |
| name              | MySubnet                             |
| network_id        | 21ed933c-323d-4708-930c-d5f82c507430 |
| project_id        | d5233415ee0b467baec14cbd2d0e1331     |
| revision_number   | 2                                    |
| segment_id        | None                                 |
| service_types     |                                      |
| subnetpool_id     | None                                 |
| tags              |                                      |
| updated_at        | 2018-03-23T23:50:11Z                 |
+-------------------+--------------------------------------+
```

To see more options when creating a subnet, use the following command:

```
openstack subnet create --help
```

Creating a router

To create a router, use the following command:

```
openstack router create <router_name>
```

Your output should look like:

```
+-------------------------+--------------------------------------+
| Field                   | Value                                |
+-------------------------+--------------------------------------+
| admin_state_up          | UP                                   |
| availability_zone_hints |                                      |
| availability_zones      |                                      |
| created_at              | 2018-03-23T23:56:35Z                 |
| description             |                                      |
| distributed             | False                                |
| external_gateway_info   | None                                 |
| flavor_id               | None                                 |
| ha                      | False                                |
| id                      | 9b5d4516-804a-4c01-9016-3a27fc4197d1 |
| name                    | MyRouter                             |
| project_id              | d5233415ee0b467baec14cbd2d0e1331     |
| revision_number         | None                                 |
| routes                  |                                      |
| status                  | ACTIVE                               |
| tags                    |                                      |
| updated_at              | 2018-03-23T23:56:35Z                 |
+-------------------------+--------------------------------------+
```

Adding a router interface

A Router Interface can be added and attached to a subnet with the command:

```
openstack router add subnet <router_name> <subnet_name>
```

In addition, you can specify an *External Gateway* for your router and connect it to the `public` Network with the following command:

```
openstack router set --external-gateway public <router_name>
```

Deleting networking objects

To delete a router with an External Gateway and subnets associated to it, use the following commands:

```
openstack router unset --external-gateway <router_name>
openstack router remove subnet <router_name> <subnet_name>
openstack router delete <subnet>
openstack network delete <network_name>
```

22.3 External layer2 connections (stitching)

Chameleon provides support for sophisticated networking experiments by providing GENI-style stitching. This capability enables users to deploy networking experiments (layer 2 and layer 3) that span Chameleon and other facilities such as FABRIC. Users can create dedicated Chameleon networks directly connected to external facilities and configure custom subnets and routers for handling these external connections. This capability is essential to users interested in experimenting with wide-area networks in a controlled environment or lower-level wide-area protocols (e.g. BGP or other routing protocols) that are not typically configurable by experimenters.

Stitched Chameleon networks are connected to external facilities using one VLAN allocated from of a pool VLANs that are statically provisioned between Chameleon and the external facility. Users can allocate one of these VLANs by making a reservation using Blazar. Currently, externally stitched networks are created by Blazar when the reservation is started. There is a pool of VLANs (3300-3309) between the Chameleon CHI@UC racks to the FABRIC site at StarLight (Chicago), and VLANs (3210-3499) from CHI@TACC to FABRIC@TACC directly via fiber.

The remainder of this document describes how to stitch Chameleon experiments to external resources using FABRIC. In addition to creating the stitched networks, you will need to know how to configure subnets and routers on dynamic VLANs as described in the *Isolated network VLANs* documentation.

> **Note**
>
> Stitching to FABRIC with the Python API is shown this a Trovi Artifact

22.3.1 Configuring a stitchable network

Your first step will require creating a stitchable network. Unlike creating other networks on Chameleon, stitchable networks can only be created by first reserving a stitchable VLAN segment. Once you reserve a VLAN segment, your network will be created automatically. To reserve a segment on the appropriate external testbed make sure to include `fabric` as the `stitch_provider` in the `resource_properties` attribute. An example is provided below:

```
openstack reservation lease create --reservation \
resource_type=network,network_name=my-stitchable-network,\
resource_properties='["=","$stitch_provider","fabric"]' \
```

(continues on next page)

(continued from previous page)

```
--start-date "2022-01-01 12:00" --end-date "2022-01-02 12:00" \
my-stitchable-network-lease
```

After your stitched VLAN network is created, you will be able to query for the network to get the specific VLAN that is used by your network. Openstack refers to the VLAN as the `segmentation_id`. The following command will display the VLAN.

```
openstack network show my-stitchable-network --format value -c provider:segmentation_id
```

At this point your Chameleon network is connected to a FABRIC `facility port` at layer 2. You can now create a FABRIC slice and specify the Chameleon `segmentation_id` to use. This is a layer 2 connection and you can configure higher-level protocols, such as IP, in any way you desire.

Alternately, if you know the desired vlan_id in advance, say it's already configured on the FABRIC `facility port`, you can create your lease as follows:

```
# in this example, we happened to know 3490 was the vlan ID we wanted
openstack reservation lease create --reservation \
resource_type=network,network_name=foo_bar,\
resource_properties='["and",["=","$stitch_provider","fabric"],["=","$segment_id","3490"]]
↪' \
segment_lease_3490
```

22.3.2 Connecting stitchable isolated networks across Chameleon sites

As both CHI@UC and CHI@TACC can be stitched to FABRIC, you're able to use FABRIC to create a Layer 2 connection between nodes at the two sites.

For a complete, current walkthrough of this pattern, see the Tips and Tricks post Running LLMs on Chameleon GPUs from FABRIC via Stitch Ports, which demonstrates the full end-to-end workflow for cross-testbed experiments and links to a companion Trovi artifact you can fork and run yourself.

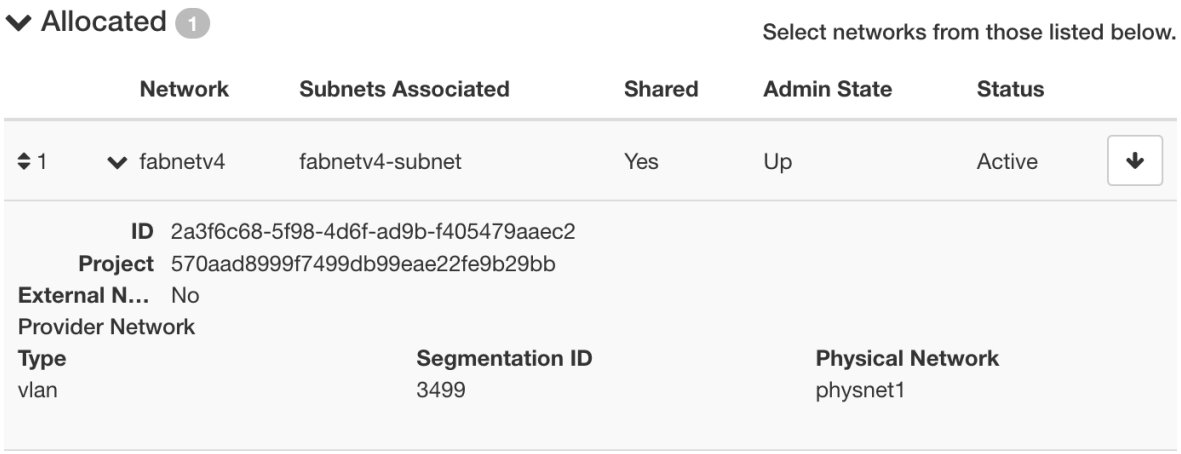
22.4 External layer 3 connectivity

In addition to configuring networks and floating IPs within a given site (see *Basic networking*), we provide a shared network named "fabnetv4", which can route traffic via the FABRIC testbed's layer3 experimental network. Nodes attached to this network may send traffic to "fabnetv4" on other Chameleon sites, or to anythign in the FABNET IPv4 address space, without needing public IPs or traversing a Neutron router.

Besides being easier to set up, this provides lower latency, higher bandwidth, and overall more flexiblity for multi-site experiments.

Site	Current status	Fabnet Router IP
UC	Available	10.191.128.1/23
TACC	Available	10.191.130.1/23

At sites where the feature is available, you will be able to see the *fabnetv4* network in Horizon,



Or by executing:

```
OS_CLOUD=tacc openstack network show fabnetv4 --fit-width
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | UP                                   |
| id                        | 2a3f6c68-5f98-4d6f-ad9b-f405479aaec2 |
| name                      | fabnetv4                             |
| project_id                | 570aad8999f7499db99eae22fe9b29bb     |
| provider:network_type     | vlan                                 |
| provider:physical_network | physnet1                             |
| provider:segmentation_id  | 3499                                 |
| router:external           | Internal                             |
| shared                    | True                                 |
| status                    | ACTIVE                               |
| subnets                   | 58283a6f-d834-4ba4-baf3-acfb59dc8648 |
+---------------------------+--------------------------------------+
```

You can therefore use it the same way as you would use *sharednet1*, but DHCP will provide an extra route, directing traffic towards *10.128.0.0/10* via the Fabnet router IP for the site.

To launch a server on fabnet:

```
openstack server create \
--network fabnetv4 \
..<other usual options>
```

Note

The fabnet router will not send traffic to the public internet. All traffic via floating IPs or otherwise internet bound will still traverse a neutron router at the chameleon site, as with any other isolated network.

After launching your instance, the following traceroutes demonstrate the new paths.

Tracroute from TACC to Google public DNS still traverses the Neutron rotuer:

```
cc@fabnet-v4-test:~$ mtr -n 8.8.8.8 --report
Start: 2024-03-01T00:55:07+0000
HOST: fabnet-v4-test              Loss%   Snt   Last   Avg  Best  Wrst StDev
  1.|-- 10.191.131.254             0.0%    10    0.1   0.1   0.1   0.3   0.0
  2.|-- 129.114.109.254            0.0%    10    4.5   4.3   1.1  12.8   4.4
  3.|-- 129.114.0.142              0.0%    10    0.6   6.4   0.5  33.7  12.3
  4.|-- 192.124.226.21             0.0%    10    6.0   6.1   5.8   6.9   0.3
  5.|-- 192.124.228.2              0.0%    10    6.3   6.2   6.1   6.3   0.1
  6.|-- 108.170.231.42             0.0%    10    7.3   7.4   7.2   7.6   0.1
  7.|-- 142.251.71.113             0.0%    10    6.3   6.3   6.2   6.4   0.1
  8.|-- 8.8.8.8                    0.0%    10    6.3   6.3   6.1   6.4   0.1
```

Traceroute from TACC to closest FABNET router shows a layer 2 path:

```
HOST: fabnet-v4-test              Loss%   Snt   Last   Avg  Best  Wrst StDev
  1.|-- 10.191.130.1               0.0%    10    0.6   0.7   0.5   0.7   0.1
```

Tracroute from TACC to FABNET router at STAR shows multiple hops through FABNET:

```
cc@fabnet-v4-test:~$ mtr -n 10.191.128.1 -r
Start: 2024-03-01T01:02:15+0000
HOST: fabnet-v4-test              Loss%   Snt   Last   Avg  Best  Wrst StDev
  1.|-- 10.130.158.1               0.0%    10    0.8   0.7   0.7   0.8   0.1
  2.|-- 10.130.128.158             0.0%    10    6.6   6.5   6.4   6.6   0.1
  3.|-- 10.133.128.134             0.0%    10   23.2  23.2  23.1  23.3   0.1
  4.|-- 10.191.128.1               0.0%    10   42.2  42.2  42.2  42.2   0.0
```

22.5 Jumbo frames

By default, Ethernet frames for networks created on each Chameleon site default to 1500 byte MTU (maximum transmission unit). However, all TOR switches on Chameleon are configured to allow for payloads of up to 9000 bytes. If you would like to experiment with jumbo frames on your private networks or over Layer 2 connections then follow the steps below to implement.

Note

You will not be able to send jumbo frames out over the public internet, as many commercial networks do not support jumbo frames. You also will not be able to send jumbo frames across two separate tenant networks via an OpenStack router. However, traffic between your nodes or over a stitched layer2 network like ExoGENI can all utilize jumbo frames.

Important

Do not set your MTU value greater than 9000. MTUs greater than 9000 bytes are not supported on Chameleon.

22.5.1 Enabling jumbo frames when creating a network

Enabling jumbo frames on a new network will ensure that the first Ethernet interface on all newly created baremetal instances will have its MTU set to the value specified.

```
openstack network create --provider-network-type vlan --mtu 9000 \
    --provider-physical-network physnet1 <network_name>
```

Note

You can verify the MTU is correct on your instance with the command `ifconfig eno1`. The first Ethernet interface is typically `eno1` for most Chameleon base images.

22.5.2 Enabling jumbo frames on existing network

You can also modify the MTU of an existing network using the command below. Note that this will only affect newly created bare metal instances.

```
openstack network set <network_name> --mtu 9000
```

22.5.3 Enabling jumbo frames on existing instances

Setting the MTU on your Chameleon network only affects instances on boot to set the first Ethernet interface. If you already have a live baremetal instance then you can simply use the command below on the instance to set MTU manually.

```
sudo ip link set dev eno1 mtu 9000
```

CHAPTER

TWENTYTHREE

FPGAS

Attention

Our previously supported Altera FPGA nodes at CHI@UC and CHI@TACC are in the process of being decommissioned.

We are actively enhancing our FPGA capabilities with new offerings in development. **Altera nodes** previously supported on Chameleon **will be unavailable after Feb. 7, 2025**.

While these updates are in progress, users can **continue accessing** our existing **Xilinx FPGA resources using documentation below**. Our team is working on implementing improved workflows to streamline current FPGA development and deployment processes. These enhancements will provide more efficient ways to utilize this hardware.

If you have ideas or suggestions regarding the use of FPGAs on Chameleon, please reach out to us at contact@chameleoncloud.org.

We will provide additional updates as new features and hardware become available. Stay connected with our platform for the latest announcements.

23.1 Introduction

Chameleon provides access to two Xilinx Alveo U280 FPGA nodes, both of which are available at CHI@UC. Each node is equipped with:

- **Memory:**
  - 32 GB DDR4 memory (two 16 GB channels)
  - 8 GB HBM (High Bandwidth Memory)
  - On-chip PLRAM memory
- **Connectivity:**
  - PCIe Gen3 x16 interface
  - Two QSFP28 ports supporting up to 100Gbps network access (*Note: These ports are not configured currently but will be available in a future update*)
- **Additional Specifications:**
  - 300 MHz default clock
  - Three Super Logic Regions (SLRs) with dedicated compute and memory resources

The workflow for using these FPGAs on Chameleon consists of four main steps:

1. Reserve a bare metal node with FPGA hardware
2. Launch an instance using a supported base image
3. Install required Xilinx software tools in your environment
4. Load your pre-compiled bitstream onto the FPGA and run your application

Important

Chameleon does not provide FPGA compilation services or development tools. Users need to compile their code elsewhere before running it on Chameleon's FPGAs. We are currently exploring new ways to provide FPGA development tools and workflows in the future.

23.2 Reserving FPGA hardware

CHI@UC provides two *compute_cascadelake_r* nodes equipped with Xilinx Alveo U280 FPGAs:

- Node `P3-CPU-038` (UUID: `89e48f7e-d04f-4455-b093-2a4318fb7026`)
- Node `P3-CPU-042` (UUID: `926a9c99-3b27-45a7-818c-e6525b9ce89c`)

To ensure you get the correct hardware, reserve these nodes specifically by UUID, as explained here in our documentation.

23.3 Launching your instance

After your reservation becomes active:

- Launch an instance using a supported upstream image for the Xilinx Runtime. Chameleon suggests using our supported CC-Ubuntu 24.04 as a base image. Advanced users may choose to use any upstream image that Xilinx supports. Find a list of supported upstream images here
- Connect to your instance via SSH

23.4 Installing Xilinx tools

Compiling code for FPGAs requires the Xilinx Vitis™ software platform, which provides a comprehensive development environment for creating FPGA-accelerated applications. The Vitis platform includes the Vitis Unified Software Platform, Vitis Core Development Kit, and Vitis AI Development Kit.

Flashing the FPGA with your bitstream requires the Xilinx Runtime (XRT) tools, which are part of the Vitis platform. The XRT tools provide a command-line interface for managing FPGA devices, including programming the FPGA with your bitstream. You can also install the XRT environment separately from the Vitis platform, although functionality may be limited.

Guidelines for installing the Vitis platform can be found in the AMD documentation. The installation requirements for Ubuntu are also provided in the documentation here.

Guidelines for installing the Xilinx Runtime (XRT) tools can be found in the XRT documentation.

23.5 Loading your bitstream

After installing the required tools, you can program the Xilinx Alveo U280 FPGA with your pre-compiled bitstream using the Xilinx Runtime (XRT) tools. The steps below are the basic workflow for flashing, but we encourage users to review the AMD documentation for more detailed instructions.

1. Verify the FPGA Device

First, ensure that the FPGA device is recognized and ready. Use the `xbmgmt` tool to examine the device status:

```
sudo xbmgmt examine --verbose
```

Look for the device BDF (Bus:Device.Function) identifier, which you'll need in the next steps. For example, it might be `0000:81:00.0`.

2. Program the FPGA with Your Bitstream

Use the `xbmgmt` tool to program the FPGA with your bitstream. Replace `<device_BDF>` with your device's BDF and `<path_to_your_bitstream>` with the path to your `.xclbin` file:

```
sudo xbmgmt program --device <device_BDF> --base --image <path_to_your_bitstream>
```

For example:

```
sudo xbmgmt program --device 0000:81:00.0 --base --image /path/to/your_bitstream.xclbin
```

This command will program the FPGA with your specified bitstream.

3. Reboot the System

After programming the FPGA, it's recommended to perform a cold reboot to ensure the new image is properly loaded:

```
sudo reboot
```

4. Verify the New Configuration

Once the system restarts, verify that the new configuration is active:

```
sudo xbmgmt examine --verbose
```

Ensure that the device is ready and the new platform UUID matches your programmed bitstream.

Important

- Ensure that your bitstream (`.xclbin` file) is compatible with the Alveo U280 FPGA.
- The `xbmgmt` tool is part of the XRT installation and is used for managing FPGA devices.
- For detailed instructions and troubleshooting, refer to the XRT documentation.
- Additional AMD instructions for bringing up and validating your card here.

CHAPTER

TWENTYFOUR

KVM

OpenStack is an Infrastructure as a Service (IaaS) platform that allows you to create and manage virtual environments. Chameleon provides an installation of OpenStack Xena using the KVM virtualization technology at the KVM@TACC site. Since the KVM hypervisor is used on this cloud, any virtual machines you upload must be compatible with KVM.

Note

For container-based edge computing on devices such as Raspberry Pis and Jetson Nanos, see CHI@Edge (docs) instead. If you're unsure whether KVM or *Bare Metal Instances* is the better fit for your experiment, see the Tips and Tricks post Bare Metal or KVM? Which Should You Choose and When.

This documentation provide basic information about how to use the OpenStack web interface and provides some information specific to using OpenStack KVM on Chameleon. The interface is similar to the bare metal sites, e.g., CHI@TACC and CHI@UC. However, the resources that you are using are virtual, rather than being tied to physical nodes. Familiarity with some concepts, such as *Key pairs* are also required for KVM.

24.1 Hardware

Instances on KVM@TACC are virtualized, but the underlying hardware may be of interest. For general compute flavors (flavors that start with "m1"), the underlying hardware is:

Component	Details
Processor	Intel Xeon E5-2670 v3 @ 2.30GHz
Chassis	Dell PowerEdge R630
Network Adapters	2 x 1 Gbps

For GPU flavors, (those starting with "g1.h100"), instances will be scheduled on the following hardware:

Component	Details
Processor	Intel Xeon Platinum 8468 @ 2.1GHz, 48C/96T, 16GT/s, 105M Cache
Memory	1 TB DDR5-4800 RAM
Chassis	Dell PowerEdge XE9640
GPU	4 x NVIDIA HGX H100 4-GPU SXM 94GB HBM2e 700W GPU
Network Adapters	1 x 25 Gbps,

Two of the H100 nodes on KVM@TACC support NVIDIA **Multi-Instance GPU (MIG)** partitioning, which splits a single H100 into up to 7 smaller, isolated GPU instances instead of allocating full-GPU passthrough. This raises the

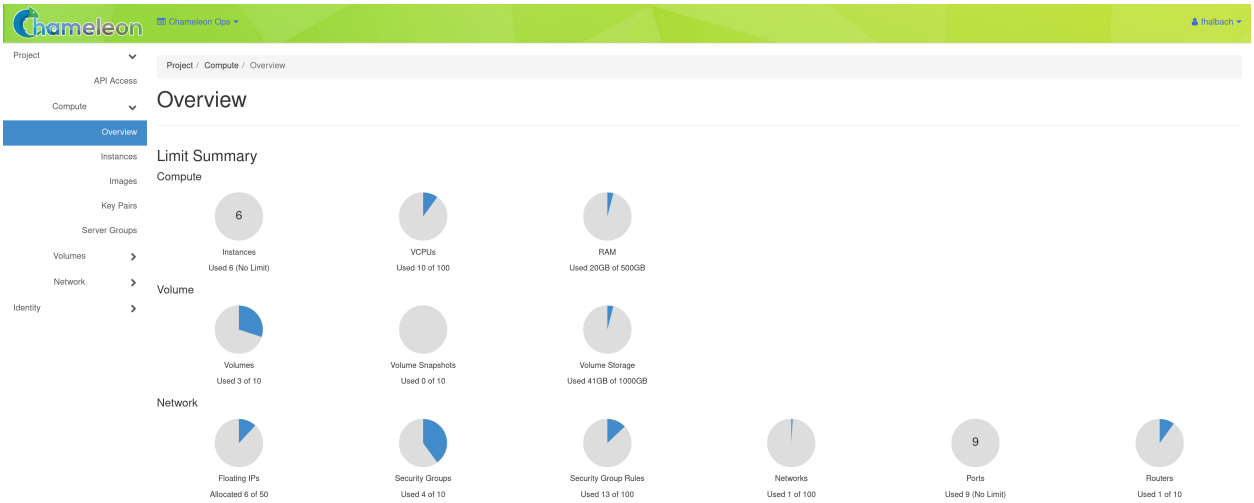
effective per-node capacity from 4 to up to 28 instances, making it easier to get GPU access for smaller jobs that do not need a full H100.

> **Attention**
>
> Standard KVM@TACC instance leases can run for up to 6 months, but leases for GPU-attached flavors (`g1.h100.*`) are capped at **7 days (1 week)** — the same duration limit that applies to bare metal host leases. This timeboxing keeps GPU capacity available to more users over time. See the Chameleon FAQ for the authoritative policy.

24.2 Work with KVM using the GUI

An easy way to use OpenStack KVM on Chameleon is via the GUI, which is similar to the GUIs for CHI@TACC and CHI@UC. You log into the GUI using your Chameleon username and password.

After a successful log in, you will see the *Overview* page as shown below. This page provides a summary of your current and recent usage and provides links to various other pages. Most of the tasks you will perform are done via the menu on the lower left and will be described below. One thing to note is that on the left, your current project is displayed. If you have multiple Chameleon projects, you can change which of them is your current project. All of the information displayed and actions that you take apply to your current project. So in the screen shot below, the quota and usage apply to the current project you have selected and no information about your other projects is shown.



24.2.1 Creating leases for VMs

Before launching an instance, you must have an active lease.

Checking availability

Before creating a lease, you can check availability using the *Flavor Calendar*. To access the Flavor Calendar:

1. Navigate to *Project* > *Reservations* > *Leases* in the navigation sidebar.
2. Click the *Flavor Calendar* button in the top right of the Leases table.

The Flavor Calendar shows availability over time for flavors, allowing you to see when resources are available for reservation.

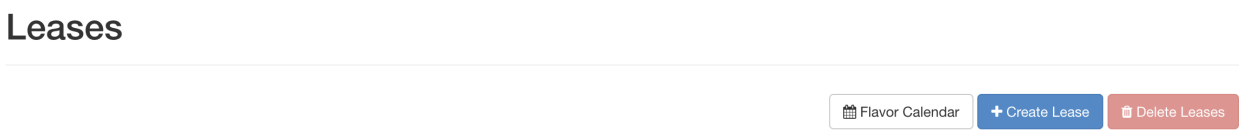


Fig. 1: The Leases table with the Flavor Calendar button highlighted

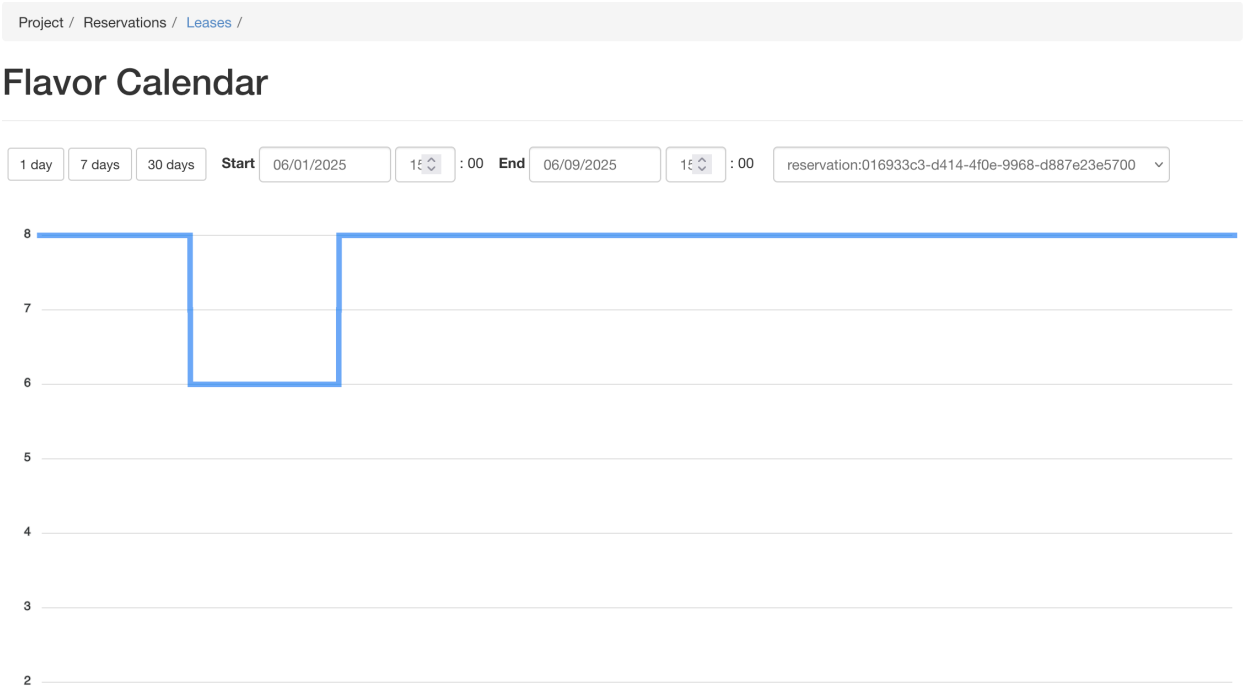


Fig. 2: The Flavor Calendar displaying flavor availability over time

Creating a lease

Once you've confirmed availability, create a lease by navigating to the *Reservations* panel and clicking *Create Lease*. In the new lease form:

- Complete the *Lease Name* and *Description* fields.
- In the *flavor* tab, select a flavor.

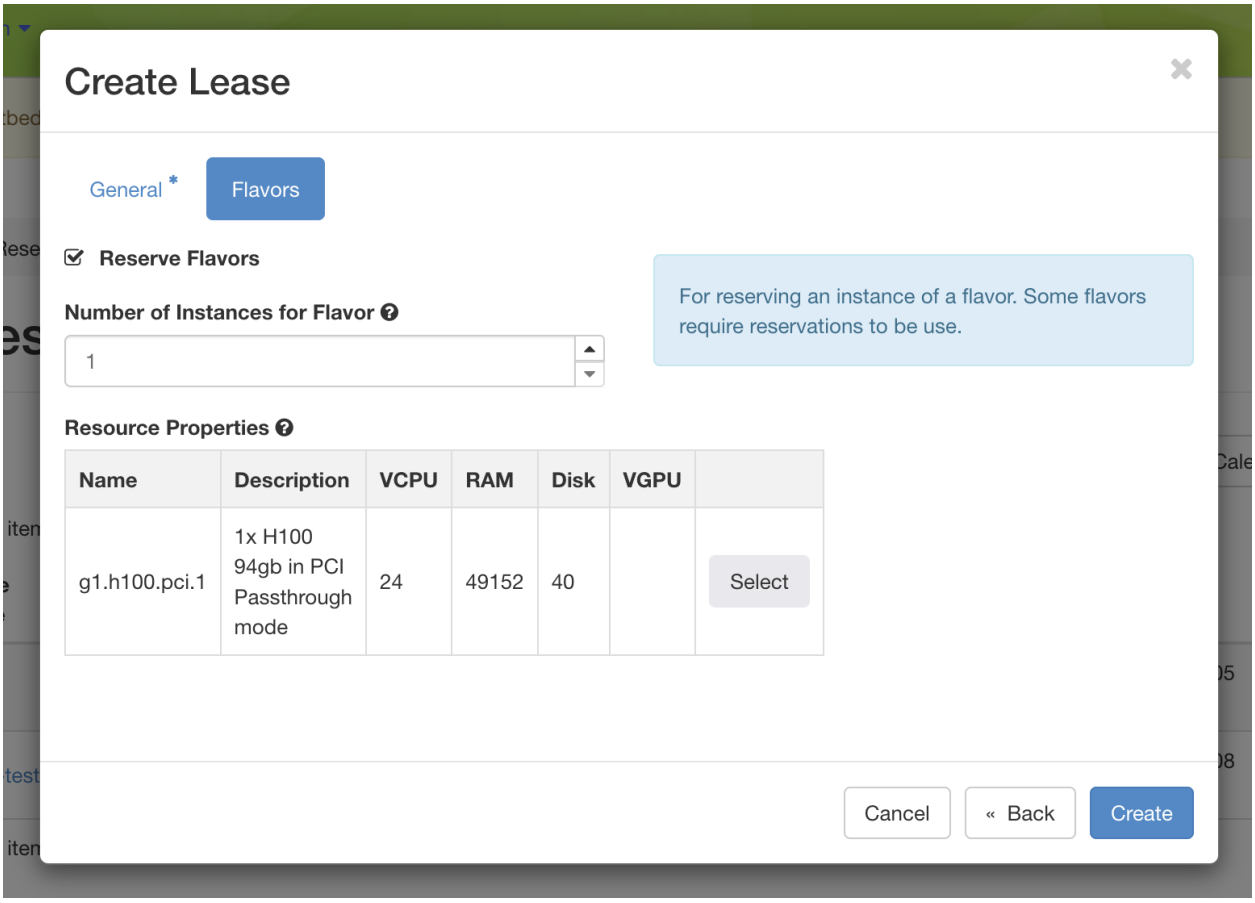


Fig. 3: Selecting a GPU-enabled flavor during lease creation

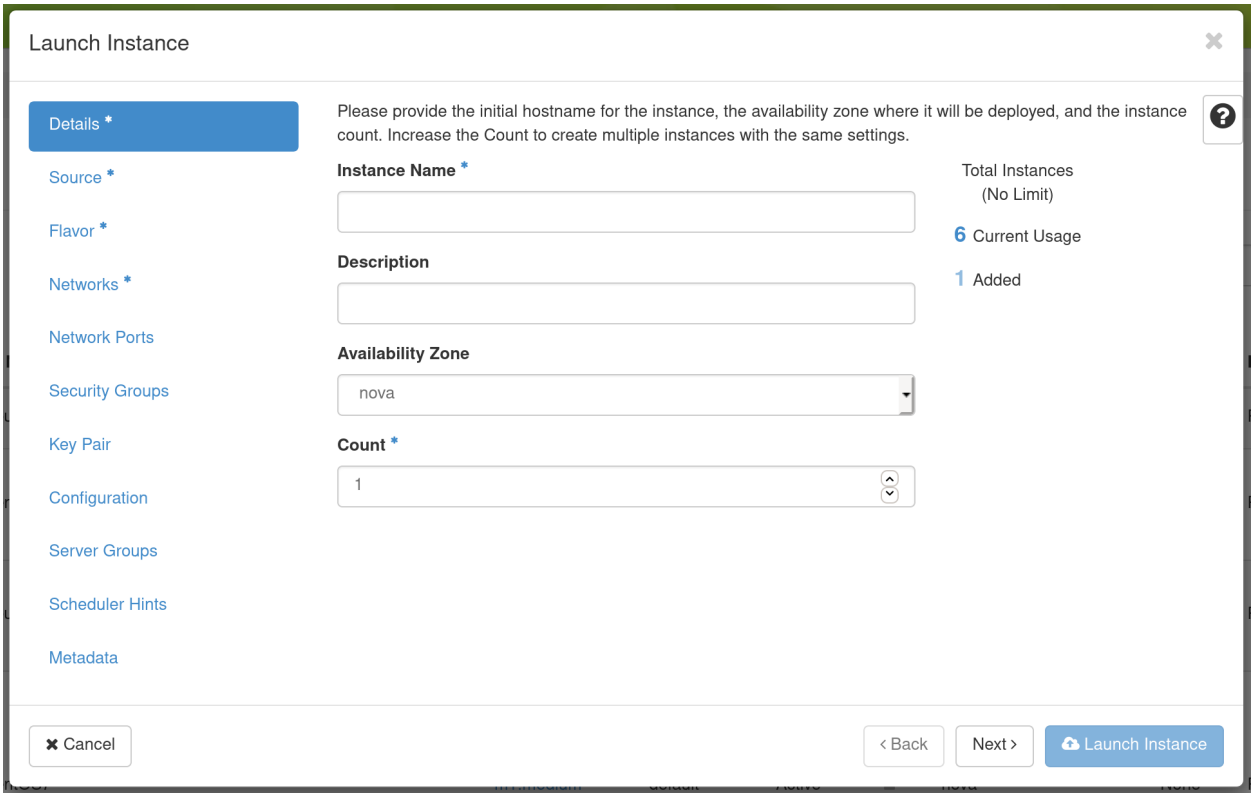
- Confirm your reservation after submitting the lease request. If capacity is insufficient, the GUI will display an error message and the lease will not be created.

> **Note**
>
> Use the Flavor Calendar to check availability before creating a lease. If you are unable to create a lease, you may need to wait for resources to become available.

> **Note**
>
> KVM@TACC lease durations follow a different policy than bare metal — see the Chameleon FAQ for details.

24.2.2 Launching instances

To launch an *Instance*, click the *Launch Instance* button. This will open the *Launch Instance* dialog.



On the *Details* tab, provide a name for this instance (to help you identify instances that you are running).

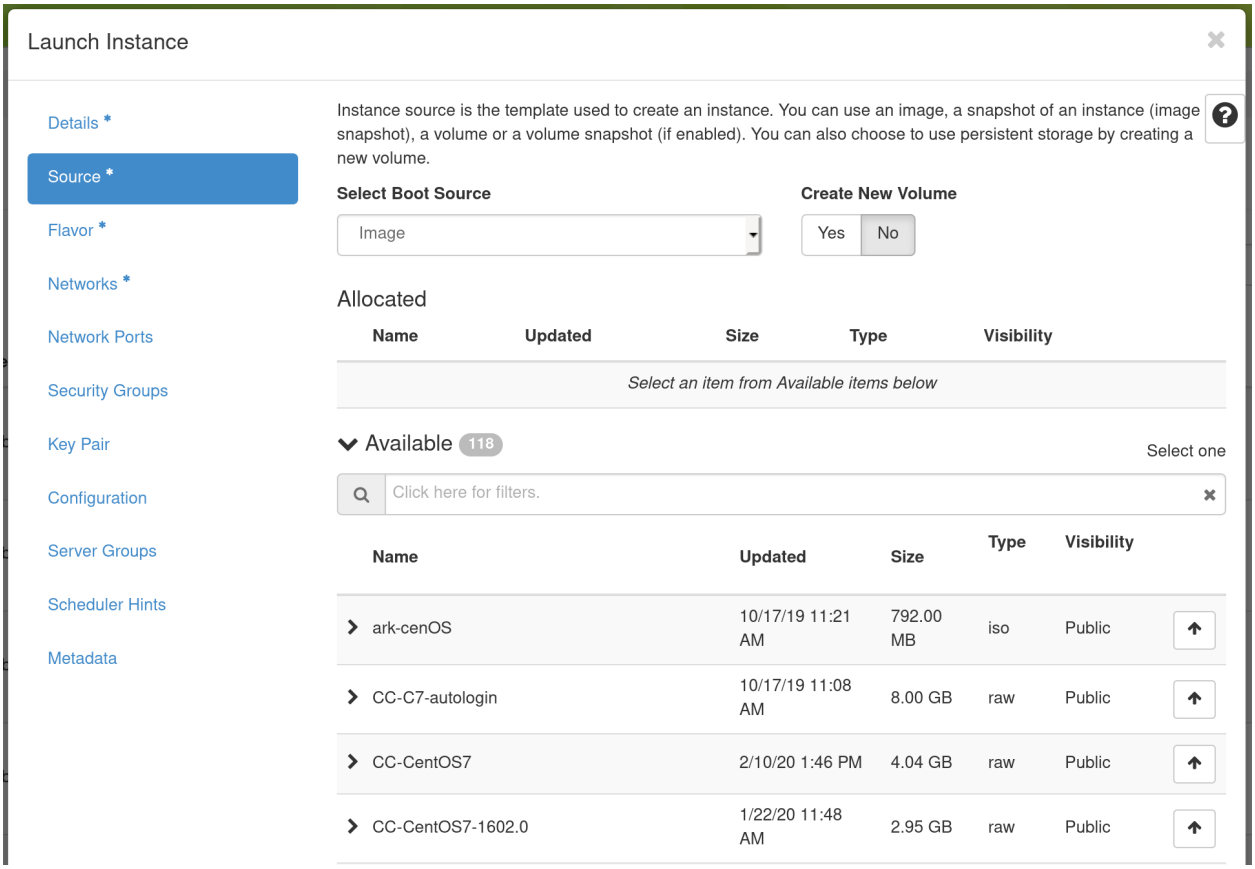
Next, go to the *Source* tab to select media to launch.

Select the *Boot Source* of the instance, which is either an *Image*, an *Instance Snapshot* (an image created from a running virtual machine), a *Volume* (a persistent virtual disk that can be attached to a virtual machine), or a "Volume Snapshot". If you select "Image" as the *Boot Source*, the *Image Name* dropdown presents a list of virtual machine images that we have provided, that other Chameleon users have uploaded and made public, or images that you have uploaded for yourself. If you select *Boot from snapshot*, the *Instance Snapshot* dropdown presents a list of virtual machine images that you have created from your running virtual machines.

Go to the *Flavor* Tab and select the amount of resources (Flavor) to allocate to the instance.

Flavors refer to the virtual machine's assigned memory and and disk size. Different images and snapshots may require a larger Flavor. For example, the `CC-CentOS7` image requires at least an `m1.small` flavor.

If you have a lease for a GPU-enabled flavor, you will see a special flavor associated with your active lease in this menu. GPU-enabled flavors are displayed with the prefix `reservation:` followed by the lease identifier (e.g., `reservation:aa46132b-5d2a-4d13-b234-46684b02399f`). Select this flavor to launch your GPU-enabled instance. See the section on *Creating Leases for VMs with GPUs* for more information.



Tip

If you select different flavors from the Flavor dropdown, their characteristics are displayed on the right.

If your VM needs differ from our defaults, you can request a custom flavor for your project from the help desk

When you are finished with this step, go to the *Key Pair* Tab.

Select an SSH key pair that will be inserted into your virtual machine. You will need to select a key pair here to be able to access an instance created from one of the public images Chameleon provides. These images are not configured with a default root password and you will not be able to log in to them without configuring an SSH key.

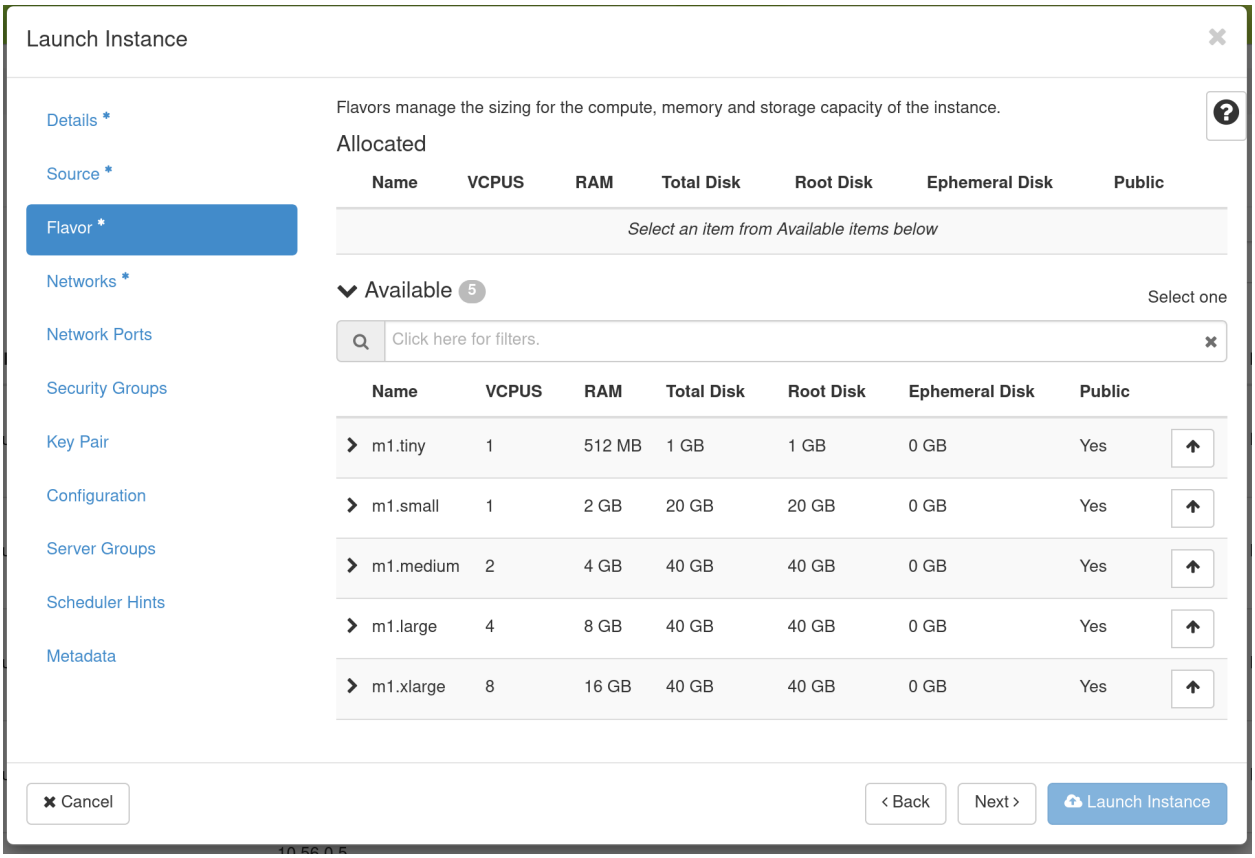
Then, go to the *Security Groups* Tab.

If you have previously defined *Security Groups*, you may select them here. Alternatively, you can configure them later.

Set up network using *Network* tab.

Select which network should be associated with the instance. Click the Up arrow next to your project's private network (PROJECT_NAME-net), not `ext-net`.

Now you can launch your instance by clicking on the *Launch* button and the *Instances* page will show progress as it starts.



24.2.3 Managing virtual machine instances

One of the main activities you'll be performing in the GUI is management of virtual machines, or instances. Go to *Project* > *Compute* > *Instances* in the navigation sidebar. For instances that you have running, you can click on the name of the instance to get more information about it and to access the VNC interface to the console. The dropdown menu to the right of the instance lets you perform a variety of tasks such as suspending, terminating, or rebooting the instance.

24.2.4 Associating a floating IP address

You may assign a Floating IP Address to your Instance by selecting *Associate Floating IP* in the dropdown menu next to your Instance on the *Instances* page.

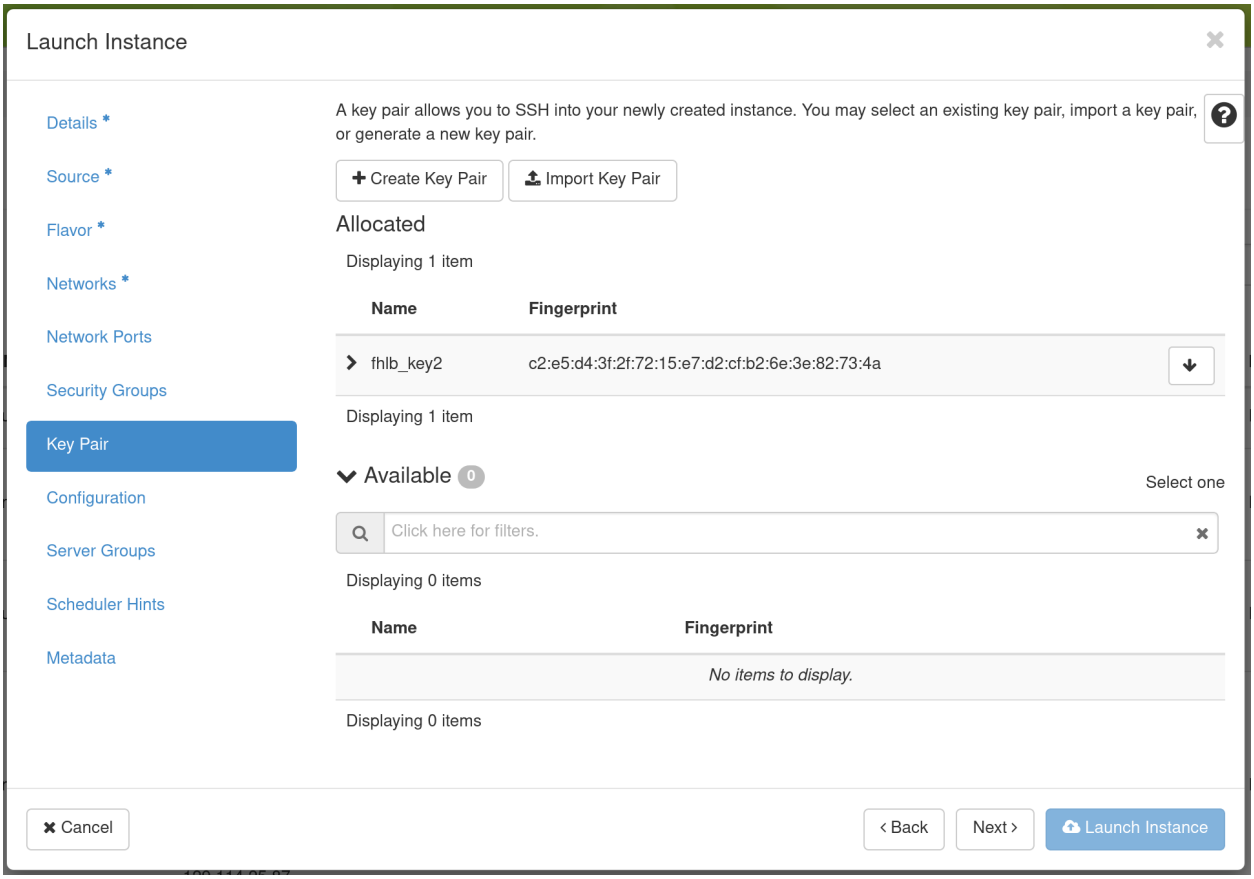
This process is similar to *Associate a floating IP* on CHI@TACC and CHI@UC bare metal sites.

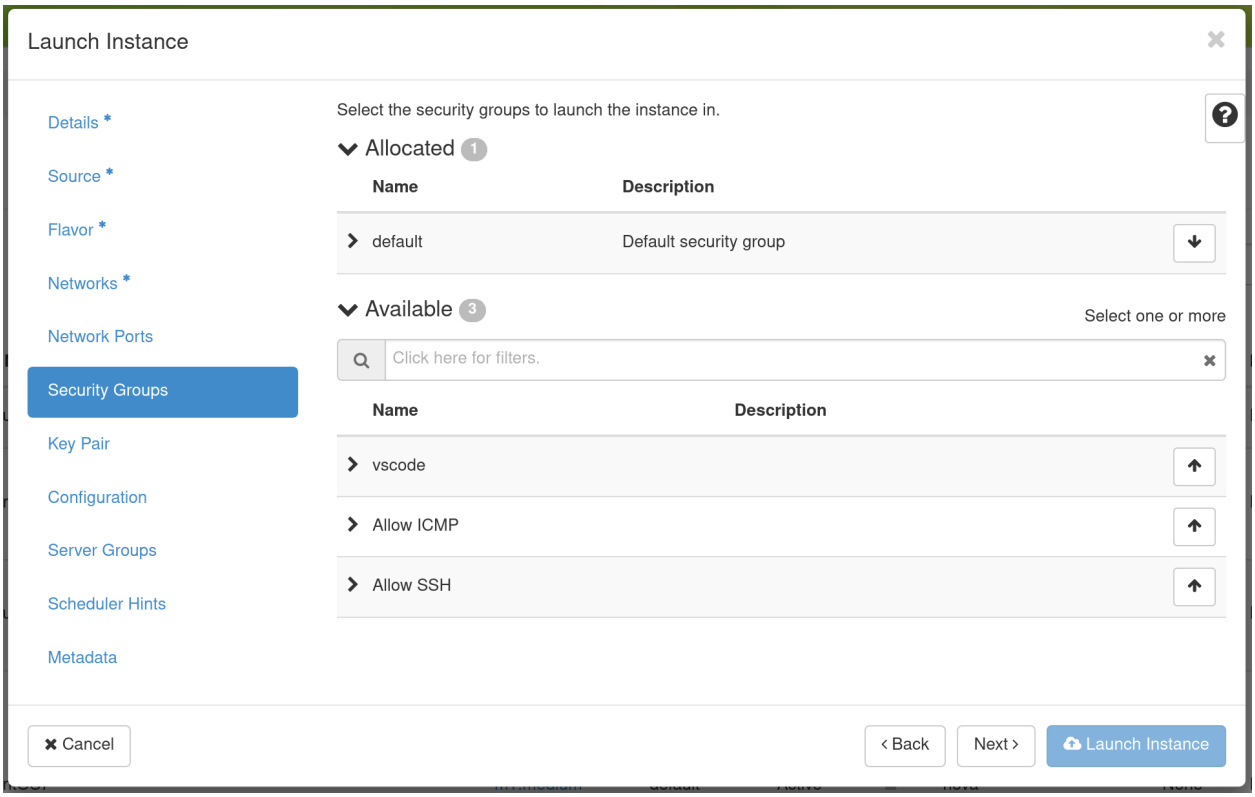
24.2.5 Key pairs

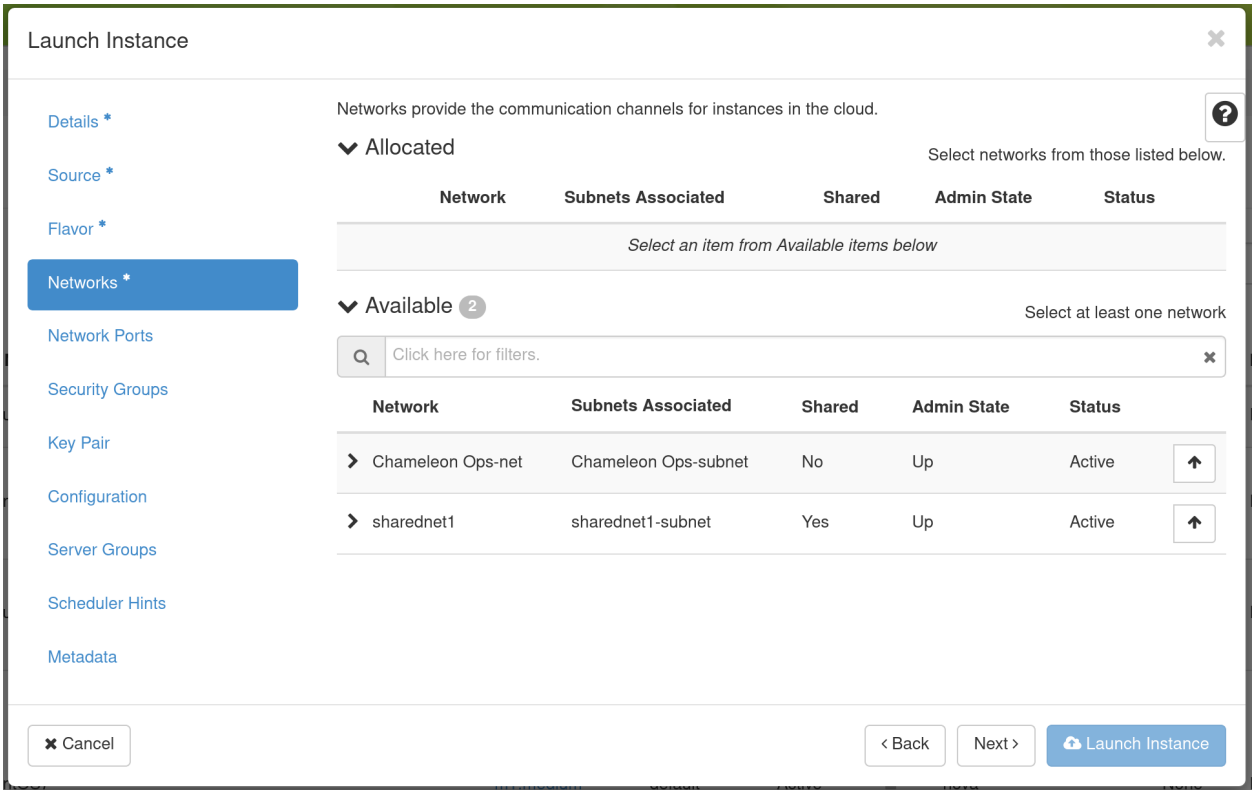
You will need to import or create SSH *Key pairs*. This process is similar to the process performed on CHI@TACC and CHI@UC bare metal sites.

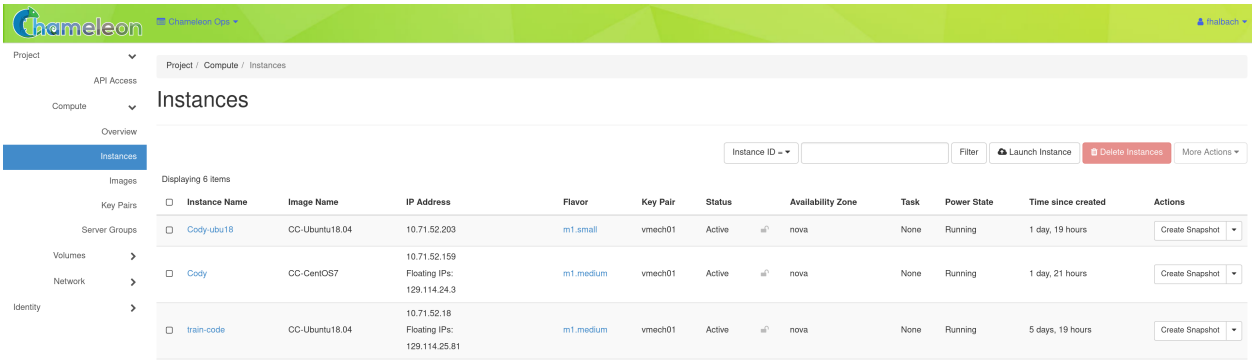
24.2.6 Security groups

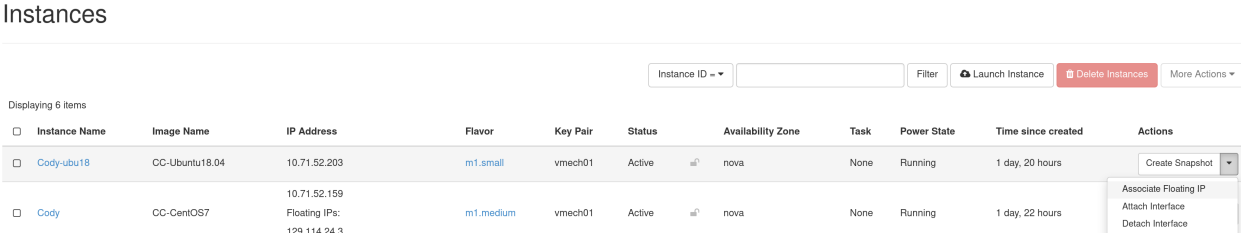
Security Groups allow you to specify what inbound and outbound traffic is allowed or blocked to Instances. Unlike the CHI@TACC and CHI@UC bare metal sites, KVM@TACC observes Security Groups for Instances.


Launch Instance
Details *
Source *
Flavor *
Networks *
Network Ports
Security Groups
Key Pair
Configuration
Server Groups
Scheduler Hints
Metadata
A key pair allows you to SSH into your newly created instance. You may select an existing key pair, import a key pair, or generate a new key pair.
+ Create Key Pair
Import Key Pair
Allocated
Displaying 1 item
Name
Fingerprint
fhlb_key2
c2:e5:d4:3f:2f:72:15:e7:d2:cf:b2:6e:3e:82:73:4a
Displaying 1 item
Available 0
Select one
Click here for filters.
Displaying 0 items
Name
Fingerprint
No items to display.
Displaying 0 items
Cancel
Back
Next
Launch Instance


Launch Instance
Details *
Source *
Flavor *
Networks *
Network Ports
Security Groups
Key Pair
Configuration
Server Groups
Scheduler Hints
Metadata
Select the security groups to launch the instance in.
Allocated 1
Name
Description
default
Default security group
Available 3
Select one or more
Click here for filters.
Name
Description
vscode
Allow ICMP
Allow SSH
Cancel
Back
Next
Launch Instance

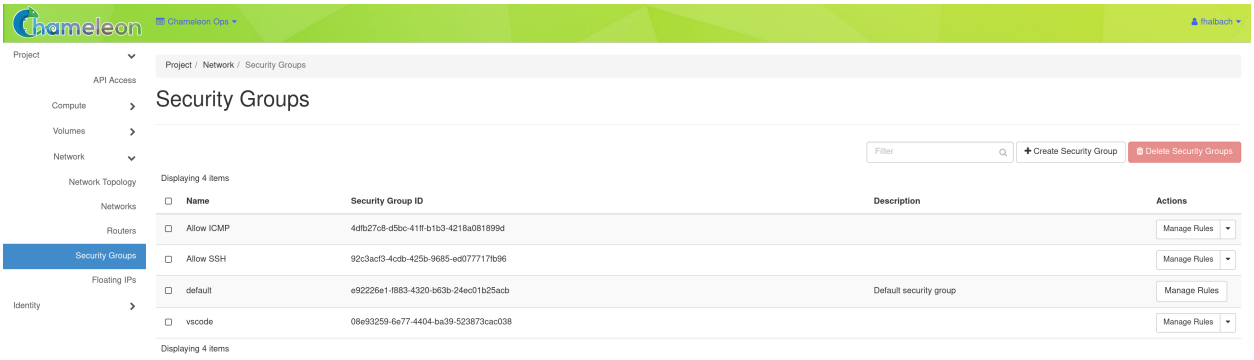

Launch Instance
Details *
Source *
Flavor *
Networks *
Network Ports
Security Groups
Key Pair
Configuration
Server Groups
Scheduler Hints
Metadata
Networks provide the communication channels for instances in the cloud.
Allocated
Select networks from those listed below.
Network
Subnets Associated
Shared
Admin State
Status
Select an item from Available items below
Available 2
Select at least one network
Click here for filters.
Network
Subnets Associated
Shared
Admin State
Status
Chameleon Ops-net
Chameleon Ops-subnet
No
Up
Active
sharednet1
sharednet1-subnet
Yes
Up
Active
Cancel
Back
Next
Launch Instance


Chameleon
Chameleon Ops
rhalbach
Project
API Access
Compute
Overview
Instances
Images
Key Pairs
Server Groups
Volumes
Network
Identity
Project / Compute / Instances
Instances
Instance ID
Filter
Launch Instance
Delete Instances
More Actions
Displaying 6 items
Instance Name
Image Name
IP Address
Flavor
Key Pair
Status
Availability Zone
Task
Power State
Time since created
Actions
Cody-ubu18
CC-Ubuntu18.04
10.71.52.203
m1.small
vmech01
Active
nova
None
Running
1 day, 19 hours
Create Snapshot
Cody
CC-CentOS7
10.71.52.159
Floating IPs:
129.114.24.3
m1.medium
vmech01
Active
nova
None
Running
1 day, 21 hours
Create Snapshot
train-code
CC-Ubuntu18.04
10.71.52.18
Floating IPs:
129.114.25.81
m1.medium
vmech01
Active
nova
None
Running
5 days, 19 hours
Create Snapshot

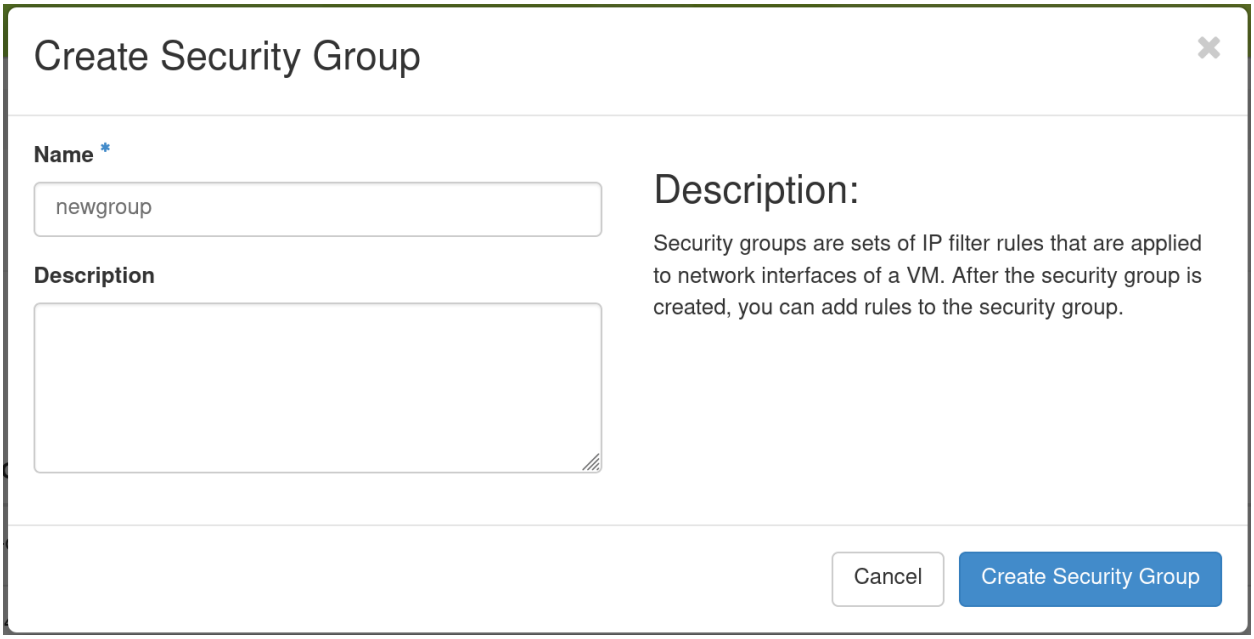

Instances
Instance ID
Filter
Launch Instance
Delete Instances
More Actions
Displaying 6 items
Instance Name
Image Name
IP Address
Flavor
Key Pair
Status
Availability Zone
Task
Power State
Time since created
Actions
Cody-ubu18
CC-Ubuntu18.04
10.71.52.203
m1.small
vmech01
Active
nova
None
Running
1 day, 20 hours
Create Snapshot
Associate Floating IP
Attach Interface
Detach Interface
Cody
CC-CentOS7
10.71.52.159
Floating IPs:
m1.medium
vmech01
Active
nova
None
Running
1 day, 22 hours

> **Note**
>
> By default, all inbound traffic is blocked to KVM@TACC Instances, including SSH. You must apply a Security Group that allows TCP port 22 inbound to access your instance via SSH.

To create a Security Group, click *Projects* > *Network* > *Security Groups* in the navigation side bar.



Click the *+Create Security Group* button to open the *Create Security Group* page.

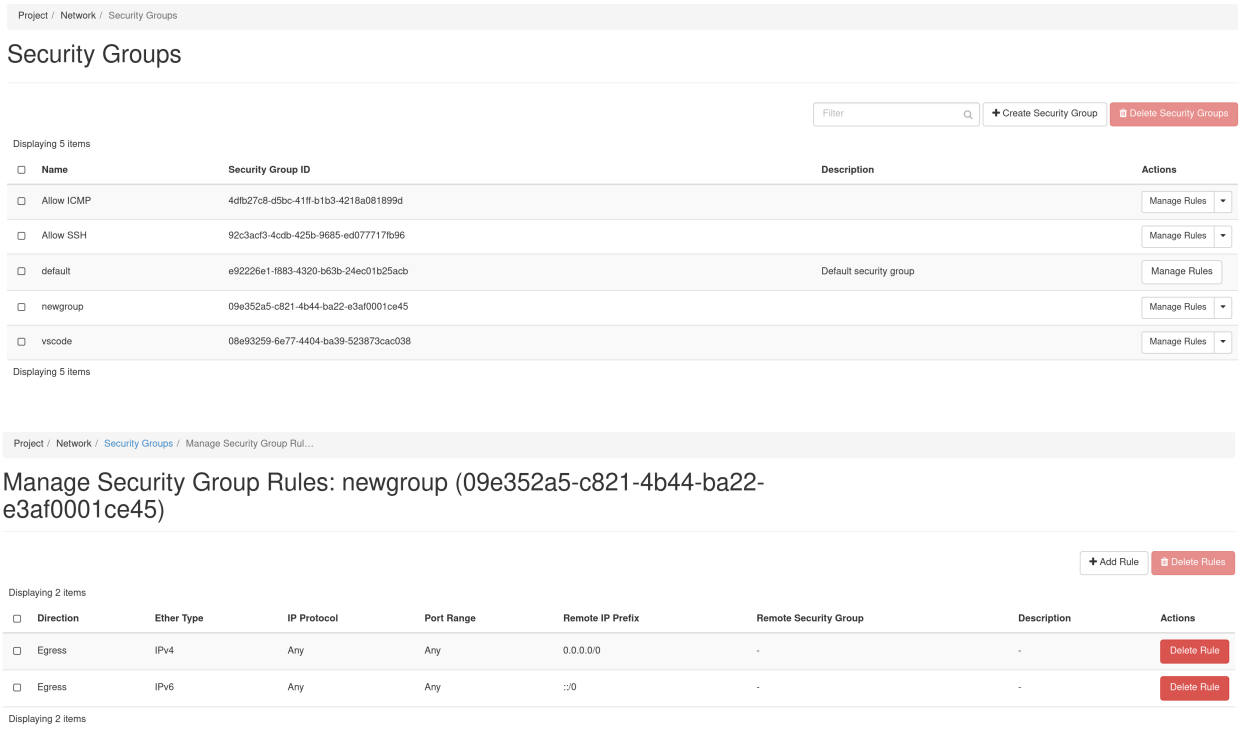


Enter a *Name* for your *Security Group*, and optionally provide a *Description*. Then click the *Create Security Group* button. Now, you should see your *Security Group* listed on the *Access and Security* page.

Click the *Manage Rules* button in the *Action* column to open the *Manage Security Group Rules* page.

The default Security Group allows outbound IPv4 and IPv6 traffic for *Any IP Protocol* and *Port Range*. If no entry for *Ingress*, no inbound traffic will be allowed. You may add an additional rule by clicking on the *+Add Rule* to open the *Add Rule* dialog.

In this dialog, you can specify *Custom TCP Rule* (or *Custom UDP Rule* or *Custom ICMP Rule*), a *Direction* (*Ingress* for inbound traffic to your Instance or *Egress* for outbound traffic) and a *Port*. Alternatively, you can use a pre-defined rule in the *Rule* dropdown, such as *SSH*. when you are finished, click *Add*.



24.2.7 Adding a security group to an instance

Once you have defined a *Security Group*, you may apply it to an Instance by clicking *Project* > *Compute* > *Instances* in the navigation sidebar and clicking the *Edit Security Groups* option in the *Actions* dropdown.

The *Security Groups* tab in the *Edit Instance* dialog will pop up.

You may click the + button next to the Security Group you wish to apply in the *All Security Groups* list on the left. Once you are finished, click *Save* to finish the process.

24.2.8 Creating a instance snapshot

Unlike the baremetal sites, where you must use the cc-snapshot tool inside your instance to create a snapshot, on KVM@TACC you can create a snapshot directly from the GUI.

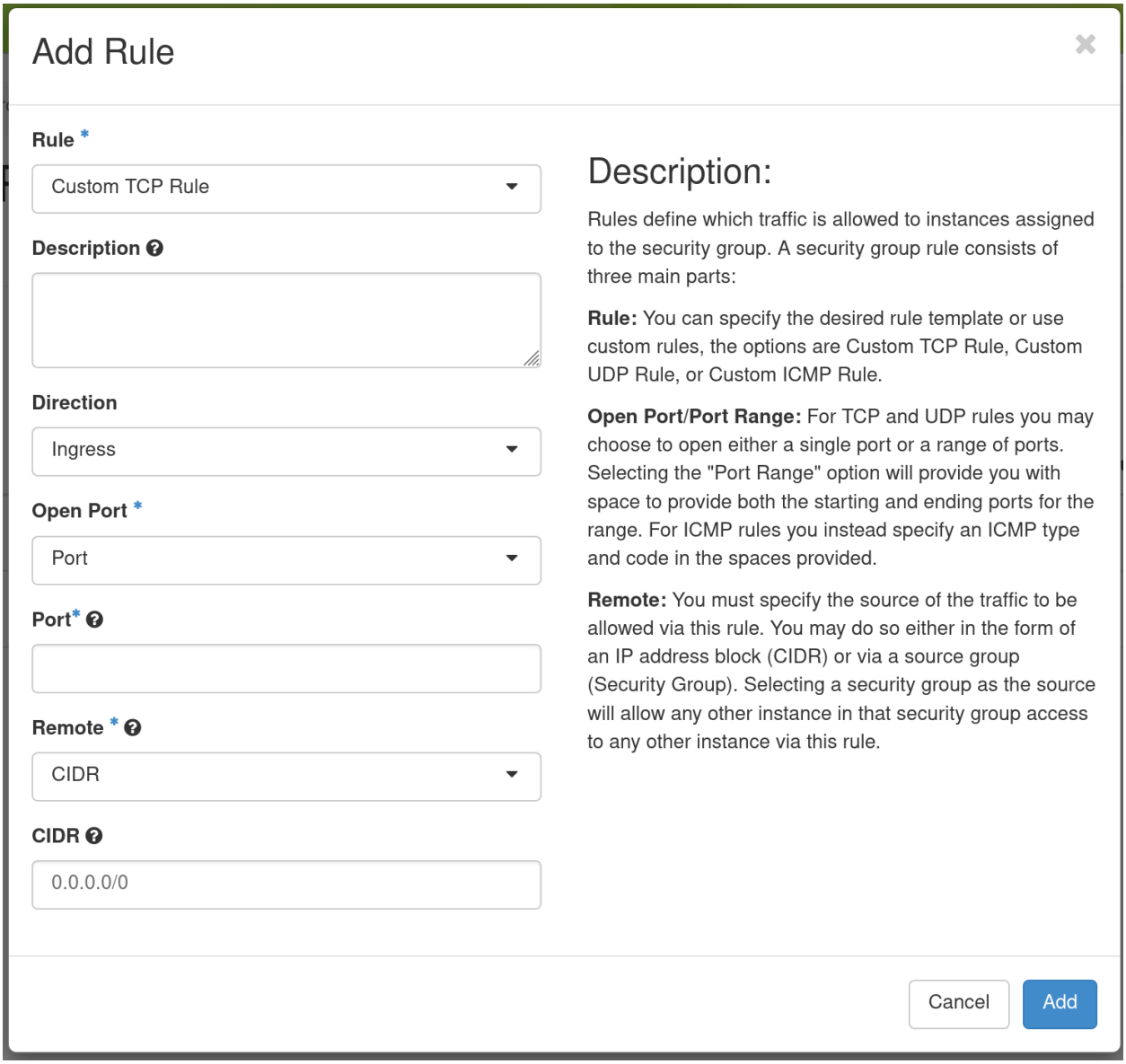
First, navigate to the Instances page by clicking "Compute > Instances" in the navigation sidebar. Click the "Create Snapshot" action next to your instance on the right size of the instance's row. Enter a snapshot name that is meaningful to you. On the Instances overview page, you'll now see a running task with the status of the snapshot job. You can see a corresponding Image in the Images page by clicking "Compute > Images" in the navigation sidebar and searching for your snapshot's name.

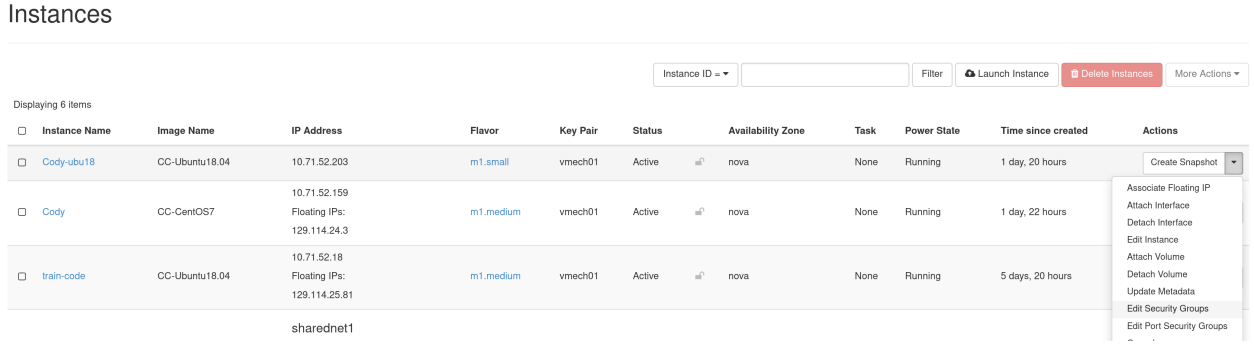
It may some time for the snapshot to complete. Once you see the Image is "Active," it is safe to delete the instance if you no longer need it.

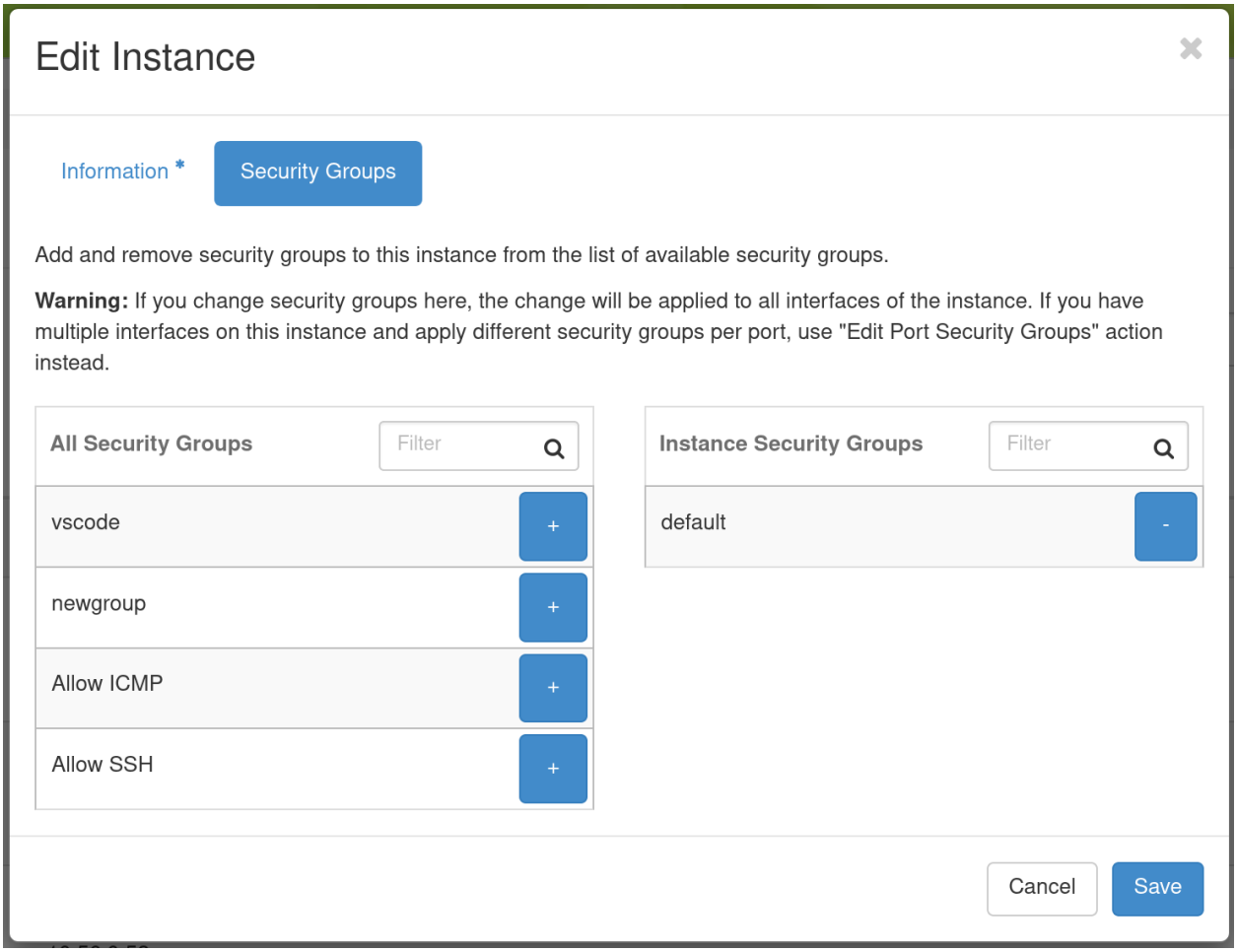
24.2.9 Launching an instance from a snapshot

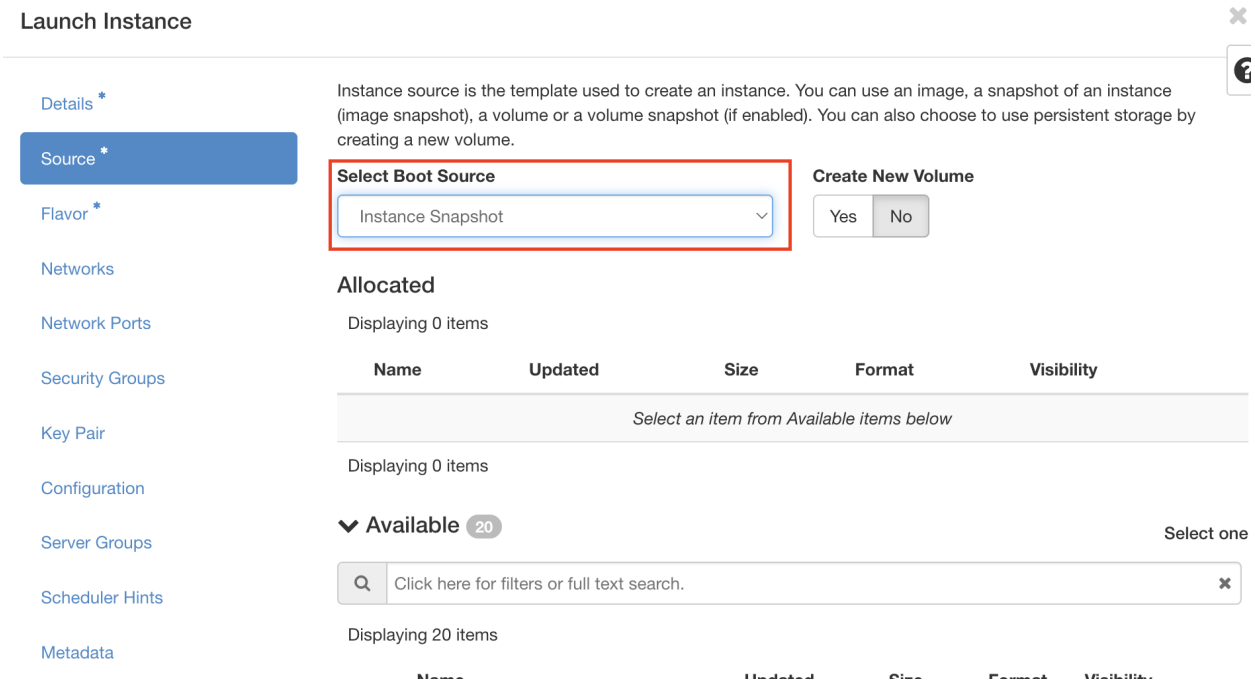
To launch an instance from a snapshot, follow the instructions from the guide above, but under the "Source" tab, select "Instance Snapshot" instead of "Image" in the dropdown. Then you can find the name of your snapshot in the table below and select it.

Alternatively, if you find your snapshot in the Images page (from "Compute > Images"), you can click "Launch" next to the snapshot's name. This will open the "Launch Instance" dialog with the snapshot preselected.


Add Rule
Rule *
Custom TCP Rule
Description
Direction
Ingress
Open Port *
Port
Port*
Remote *
CIDR
CIDR
0.0.0.0/0
Description:
Rules define which traffic is allowed to instances assigned to the security group. A security group rule consists of three main parts:
Rule: You can specify the desired rule template or use custom rules, the options are Custom TCP Rule, Custom UDP Rule, or Custom ICMP Rule.
Open Port/Port Range: For TCP and UDP rules you may choose to open either a single port or a range of ports. Selecting the "Port Range" option will provide you with space to provide both the starting and ending ports for the range. For ICMP rules you instead specify an ICMP type and code in the spaces provided.
Remote: You must specify the source of the traffic to be allowed via this rule. You may do so either in the form of an IP address block (CIDR) or via a source group (Security Group). Selecting a security group as the source will allow any other instance in that security group access to any other instance via this rule.
Cancel
Add


Instances
Displaying 6 items
Instance Name
Image Name
IP Address
Flavor
Key Pair
Status
Availability Zone
Task
Power State
Time since created
Actions
Cody-ubu18
CC-Ubuntu18.04
10.71.52.203
m1.small
vmech01
Active
nova
None
Running
1 day, 20 hours
Create Snapshot
Cody
CC-CentOS7
10.71.52.159
Floating IPs:
129.114.24.3
m1.medium
vmech01
Active
nova
None
Running
1 day, 22 hours
train-code
CC-Ubuntu18.04
10.71.52.18
Floating IPs:
129.114.25.81
m1.medium
vmech01
Active
nova
None
Running
5 days, 20 hours
sharednet1
Associate Floating IP
Attach Interface
Detach Interface
Edit Instance
Attach Volume
Detach Volume
Update Metadata
Edit Security Groups
Edit Port Security Groups


Edit Instance
Information *
Security Groups
Add and remove security groups to this instance from the list of available security groups.
Warning: If you change security groups here, the change will be applied to all interfaces of the instance. If you have multiple interfaces on this instance and apply different security groups per port, use "Edit Port Security Groups" action instead.
All Security Groups
Filter
vscode
newgroup
Allow ICMP
Allow SSH
Instance Security Groups
Filter
default
Cancel
Save


Launch Instance
Details *
Source *
Flavor *
Networks
Network Ports
Security Groups
Key Pair
Configuration
Server Groups
Scheduler Hints
Metadata
Instance source is the template used to create an instance. You can use an image, a snapshot of an instance (image snapshot), a volume or a volume snapshot (if enabled). You can also choose to use persistent storage by creating a new volume.
Select Boot Source
Instance Snapshot
Create New Volume
Yes
No
Allocated
Displaying 0 items
Name
Updated
Size
Format
Visibility
Select an item from Available items below
Displaying 0 items
Available 20
Select one
Click here for filters or full text search.
Displaying 20 items

24.3 Work with KVM using the CLI

> **Note**
>
> For general information on CLI authentication and use, see *Command Line Interface (CLI)*.

24.3.1 Creating a lease and launching an instance

KVM@TACC reservations use the same flavor-based lease pattern as the rest of the CLI reservation workflow. See *Creating a lease for a flavor (on KVM@TACC)* for creating a lease for a flavor and launching an instance against it.

24.3.2 Security groups

Unlike the bare metal sites, KVM@TACC enforces *Security Groups* on every instance, and blocks **all** inbound traffic — including SSH — by default. Before you can reach an instance over SSH, you must create a Security Group that allows inbound TCP port 22 and apply it to the instance.

Create a Security Group and add a rule allowing SSH:

```
openstack security group create allow-ssh
openstack security group rule create --protocol tcp --dst-port 22:22 allow-ssh
```

Apply it to a running instance:

```
openstack server add security group <server> allow-ssh
```

To remove a Security Group from an instance:

```
openstack server remove security group <server> allow-ssh
```

> **Tip**
>
> You can also apply Security Groups at launch time with the `--security-group` flag on `openstack server create`.

24.3.3 Creating an instance snapshot

Bare metal instances require the *The cc-snapshot utility* to create a snapshot, since Ironic-managed bare metal has no hypervisor-level snapshot support. KVM@TACC instances are virtualized, so you can snapshot them directly through Nova instead — no additional tooling required:

```
openstack server image create --name <snapshot-name> <server>
```

Once the snapshot completes, the new image appears in `openstack image list` and can be used to launch new instances the same way as any other image.

24.3.4 Uploading qcow2 images to raw format for better instance launch performance

KVM images are stored on our Ceph cluster, which is able to serve raw images much faster than qcow2 for instance launches. Openstack includes the experimental command Glance image-create-via-import, which allows uploading of images in various standard formats including qcow2 to then be automatically converted to raw in the backend.

In order to use this method, authenticate to KVM using the OpenStack RC script downloaded from the KVM@TACC site as described in *The OpenStack RC script*.

Next, issue the following command:

```
glance image-create-via-import --container-format bare --disk-format qcow2 --
↪file </path/to/image> --name <image name>
```

Details and other options for this command are available via the Glance image-create-via-import documentation.

Attention

Glance image-create-via-import is currently unable to handle conversion of iso images to raw.

Alternatively, you may convert qcow2 images to raw format before upload. qemu-img is one tool that is able to this with the following command:

```
qemu-img convert -f qcow2 -O raw <original.qcow2> <converted.img>
```

Once converted, use glance to upload the image:

```
openstack image create --file </path/to/converted.img> --disk-format raw
↪<image-name>
```

Details and other options for this command are available within Openstack documentation.

24.4 Load balancer as a service

Available on KVM@TACC is the OpenStack Octavia Load Balancer as a Service (LBaas). This service allows a single IP address to be used to distribute connections among a number of virtual machine instances. For the following description, it is assumed that there are already several virtual machines running an HTTP server on port 80, serving a page at the root path. To create a *Load Balancer*, click on *Project* > *Network* > *Load Balancers* in the navigation sidebar, then the *Create Load Balancer* button. This will open the *Create Load Balancer* dialog.

Give your load balancer a name, and select the subnet that corresponds to the one used by the virtual machines. Click *Next*, or *Listener Details*.

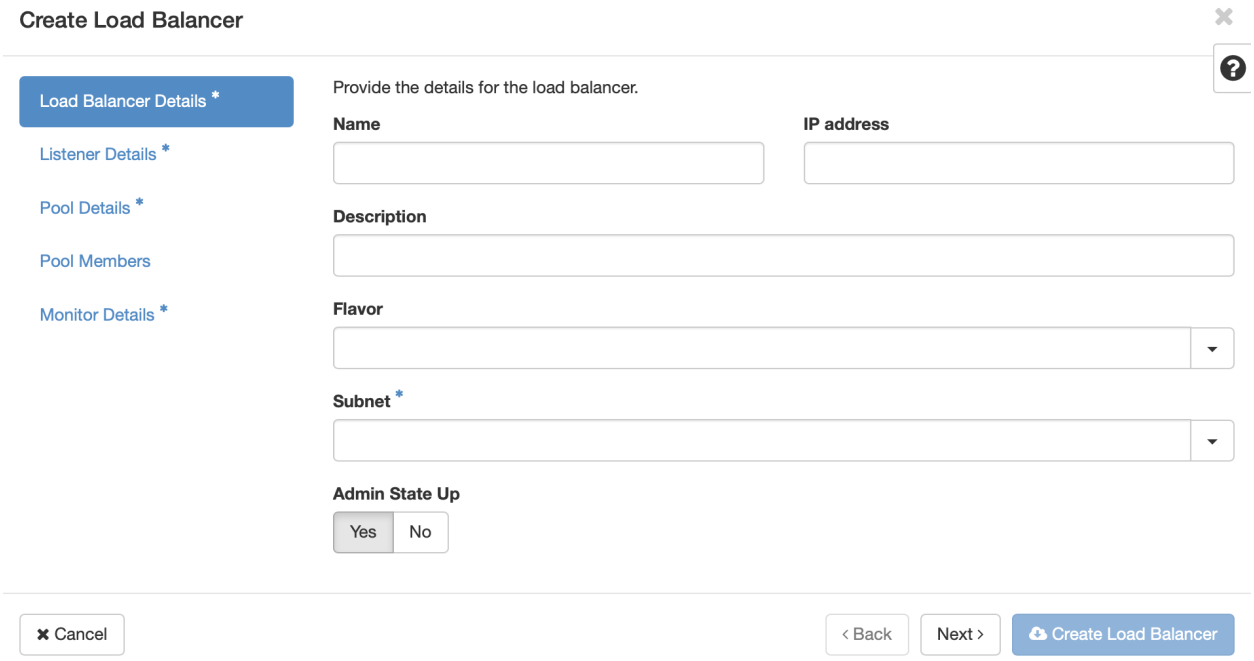
The listener is the port that will accept incoming connetions. Select the appropriate protocol for the service, in this case *HTTP*. If selecting *TCP* or *UDP* also provide the desired port. Click *Next* or *Pool Details*.

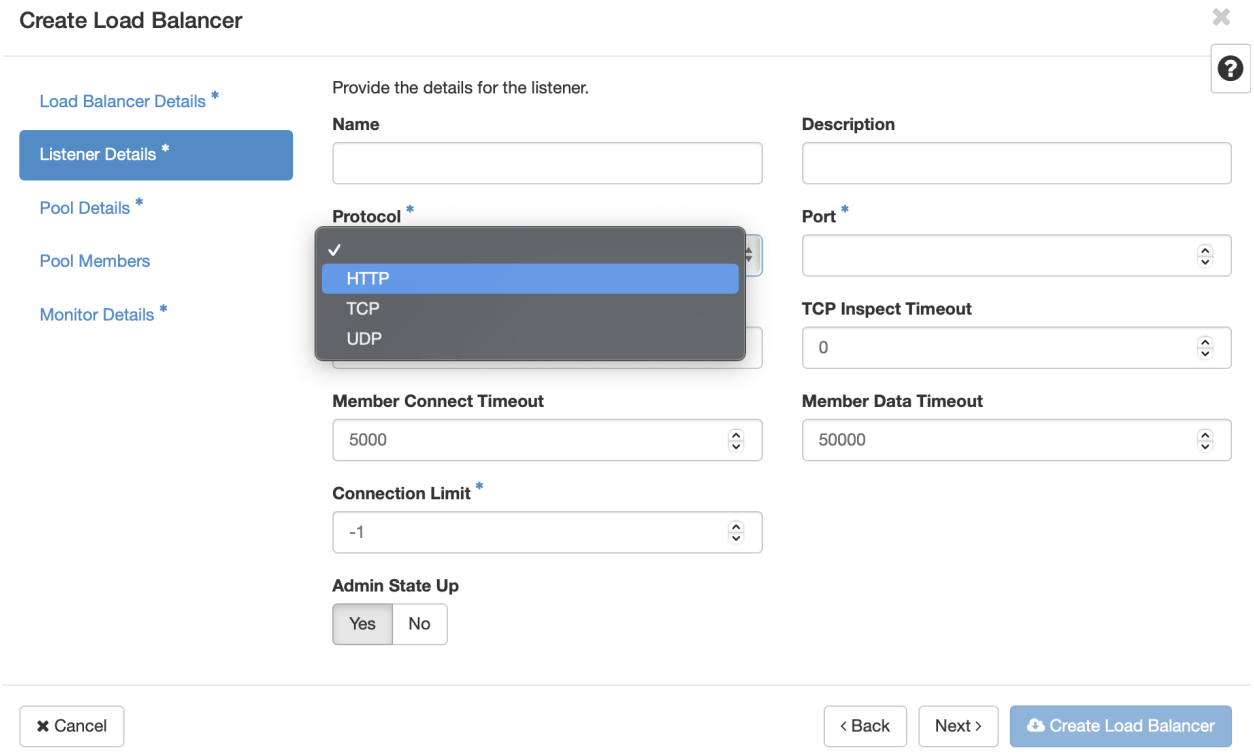
Choose the desired load balancing algorithm. This will determine the way in which the load balancer will select which VM receives incoming requests. Click *Next* or *Pool Members*.

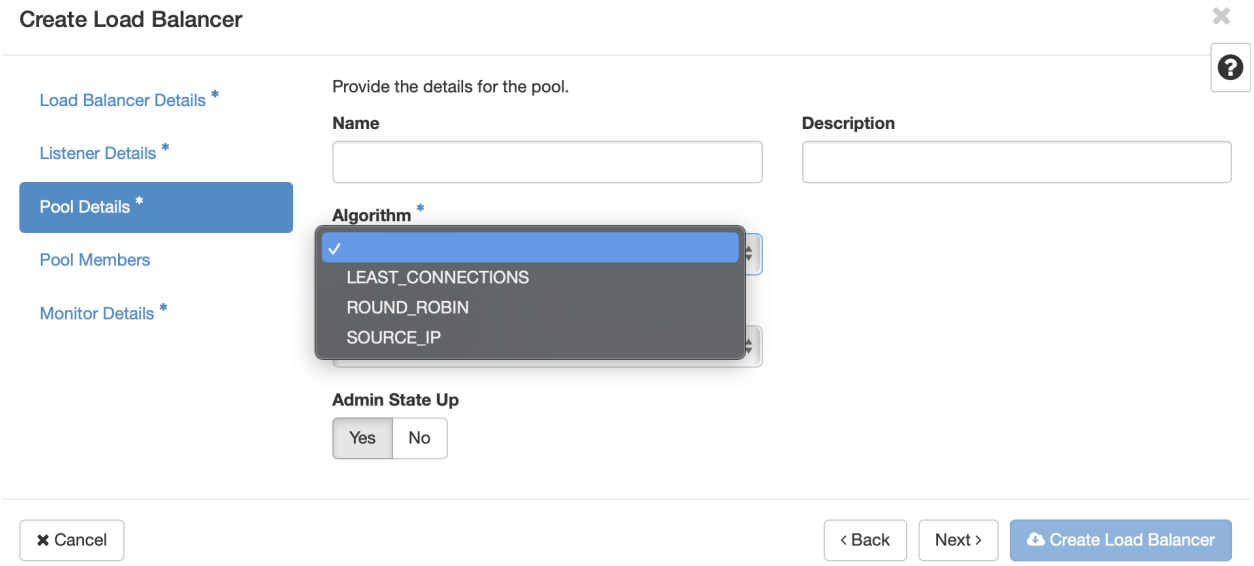
Here you will select the virtual machines that will participate in the load balacing. Click the *Add* button next to the instances, after which their IP address and subnet will be added to the *Allocated Members* list at the top. You will need to provide the port number for the hosted service for each member. For our HTTP servers, it is port 80. This does not need to match the port of the load balancer's *listener*.

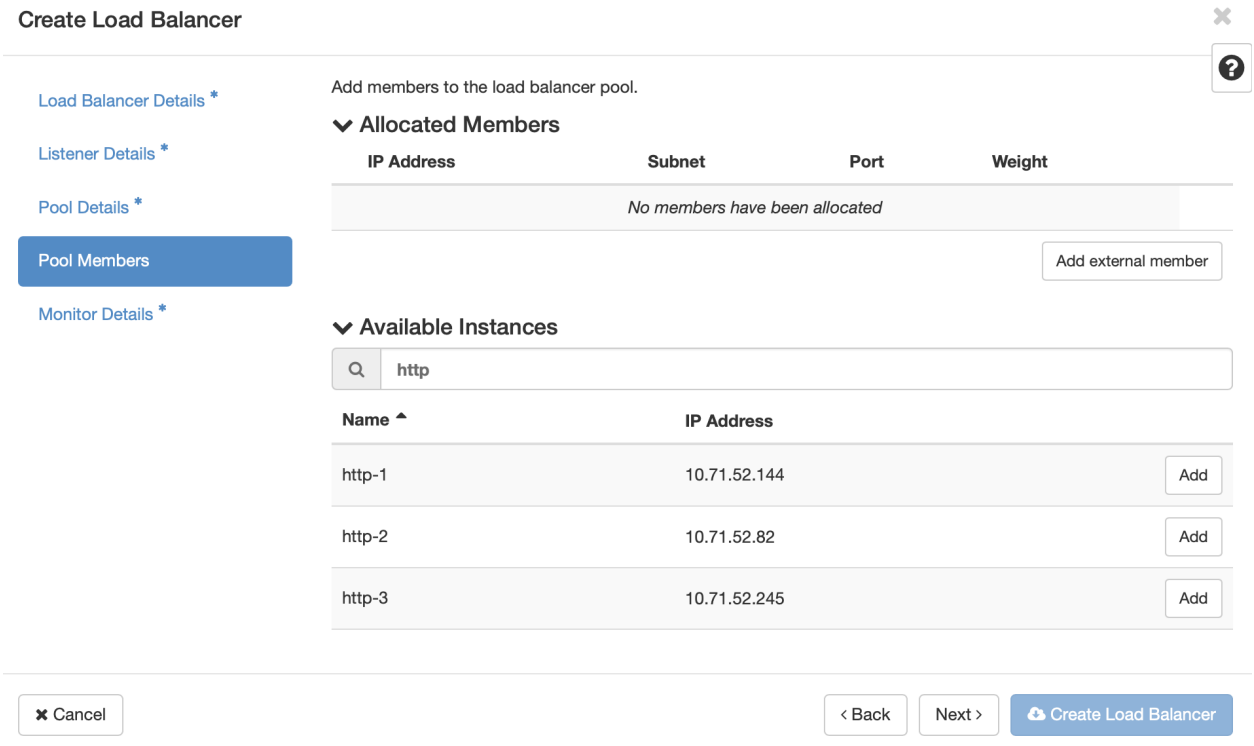
Once you've selected the pool members, click *Next* or *Monitor Details*. Here you will configure how the load balancer monitors the servies on the virtual machines to ensure that they are ready to receive traffic. In our example, selecting *HTTP* adds configuration options for *HTTP Method*, *Expected Codes*, and *URL path*. Since the HTTP services on the VMs in the *pool members* are configured to serve a page on the root path, the default values will work. Click *Create Load Balancer*

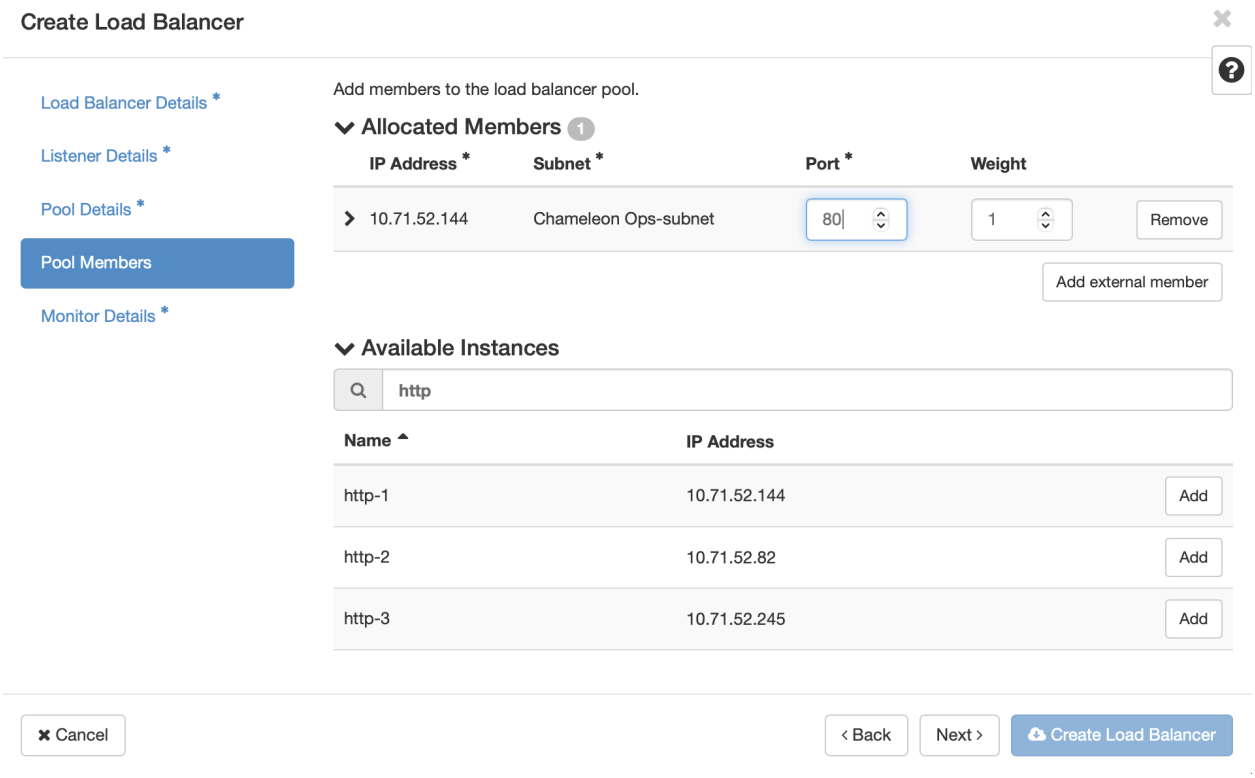
While the load balancer is being created, the dashboard will show a *Provisioning Status* of *Pending Create* . Once the process is complete, the status should be *Active*, and *Operating Status* should be *Online*. An *Operating Status*

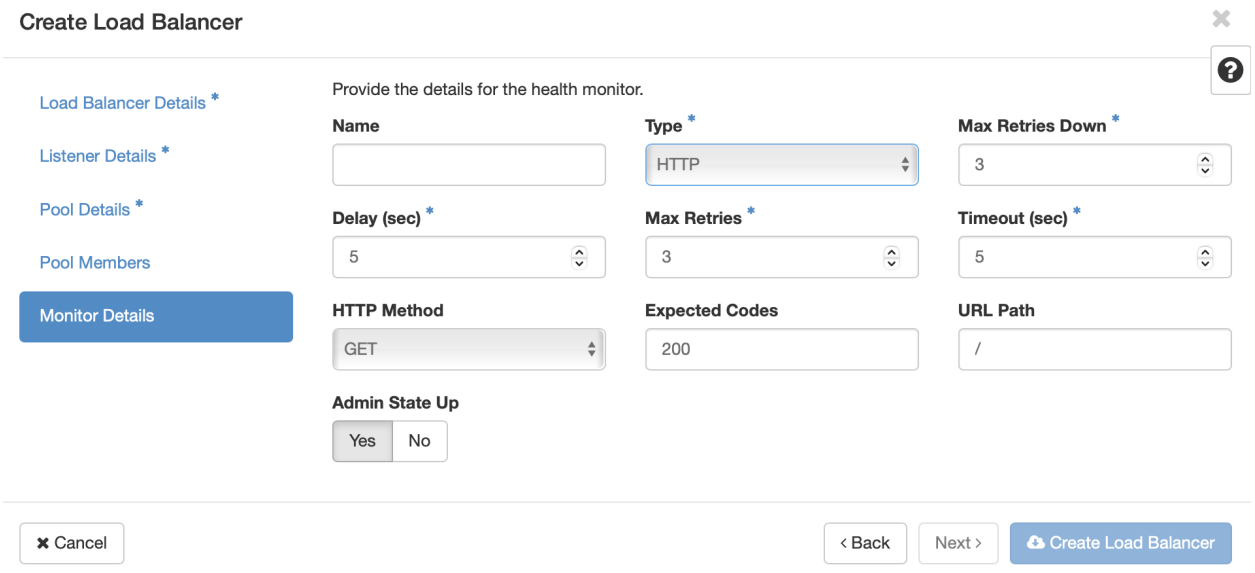

Create Load Balancer
Load Balancer Details *
Listener Details *
Pool Details *
Pool Members
Monitor Details *
Provide the details for the load balancer.
Name
IP address
Description
Flavor
Subnet *
Admin State Up
Yes
No
Cancel
Back
Next
Create Load Balancer


Create Load Balancer
Load Balancer Details *
Listener Details *
Pool Details *
Pool Members
Monitor Details *
Provide the details for the listener.
Name
Description
Protocol *
HTTP
TCP
UDP
Port *
TCP Inspect Timeout
0
Member Connect Timeout
5000
Member Data Timeout
50000
Connection Limit *
-1
Admin State Up
Yes
No
Cancel
Back
Next
Create Load Balancer

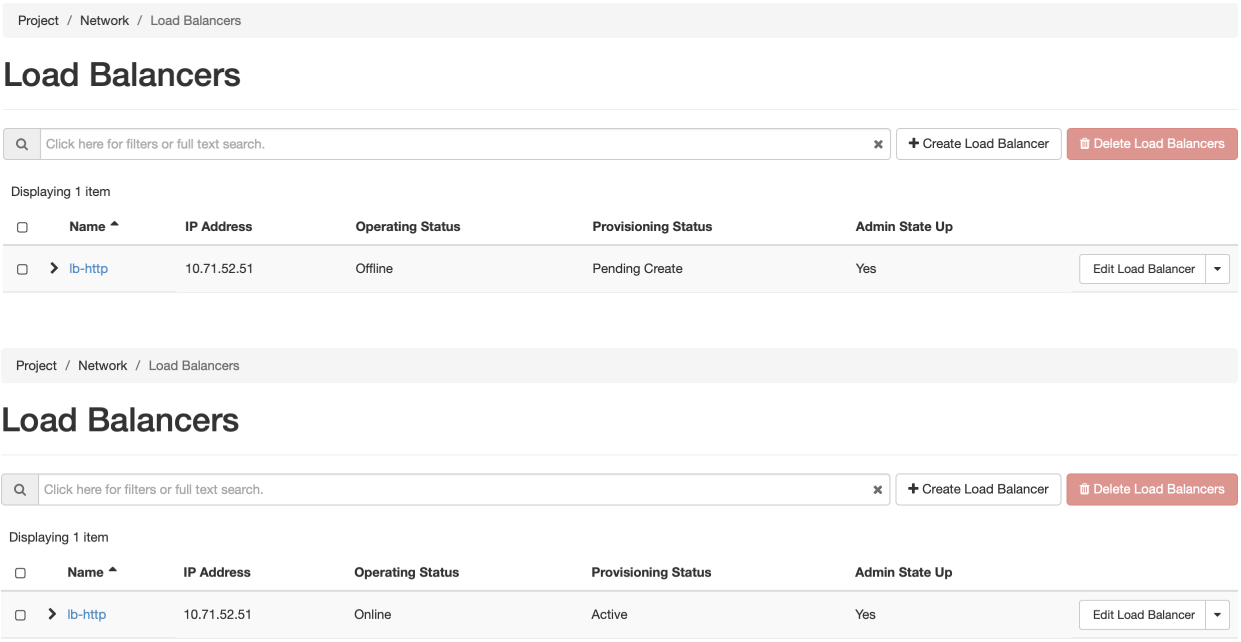

Create Load Balancer
Load Balancer Details *
Listener Details *
Pool Details *
Pool Members
Monitor Details *
Provide the details for the pool.
Name
Description
Algorithm *
LEAST_CONNECTIONS
ROUND_ROBIN
SOURCE_IP
Admin State Up
Yes
No
Cancel
Back
Next
Create Load Balancer


Create Load Balancer
Load Balancer Details *
Listener Details *
Pool Details *
Pool Members
Monitor Details *
Add members to the load balancer pool.
Allocated Members
IP Address
Subnet
Port
Weight
No members have been allocated
Add external member
Available Instances
http
Name
IP Address
http-1
10.71.52.144
Add
http-2
10.71.52.82
Add
http-3
10.71.52.245
Add
Cancel
Back
Next
Create Load Balancer


Create Load Balancer
Load Balancer Details *
Listener Details *
Pool Details *
Pool Members
Monitor Details *
Add members to the load balancer pool.
Allocated Members 1
IP Address * Subnet * Port * Weight
10.71.52.144 Chameleon Ops-subnet 80 1 Remove
Add external member
Available Instances
http
Name IP Address
http-1 10.71.52.144 Add
http-2 10.71.52.82 Add
http-3 10.71.52.245 Add
Cancel
Back Next Create Load Balancer


Create Load Balancer
Load Balancer Details *
Listener Details *
Pool Details *
Pool Members
Monitor Details
Provide the details for the health monitor.
Name
Type * HTTP
Max Retries Down * 3
Delay (sec) * 5
Max Retries * 3
Timeout (sec) * 5
HTTP Method GET
Expected Codes 200
URL Path /
Admin State Up
Yes No
Cancel
Back Next Create Load Balancer

of "*Offline*" or "*Error*" indicates that the load balancer cannot satisfy the service check specified in *Monitor Details*. Ensure that the services are running on each VM, and that they return the expected status.



You can assign a Floating IP address to the load balancer by clicking on the down arrow button next to *Edit Load Balancer*, and selecting *Associate Floating IP*. This process is similar to associatig af Floating IP to a virtual machine instnace. Making changes to the various components of the load balancer by clicking on the blue-colord name of the load balancer in the list. From here, the *listeners*, *pools*, and *health monitors* can be updated, if needed.

To learn more about how to use the Octavia Load Balancer, refer to the Basic Load Balancing Cookbook on the official OpenStack documentation

24.5 Persistent storage via volumes

KVM supports volumes, which allow you to attach persistent storage to your instances. The storage on your instance is ephemeral, meaning that it will be lost when the instance is deleted. Since instances are tied to a reservation, which expire, it may be useful to use volumes to store data that you want to keep beyond the lifetime of an instance.

Currently, KVM@TACC does not have an object store, but you can set up access to the CHI@TACC object store, which is located in the same data center. See the *Accessing object store from KVM instances* docs for KVM-specific instructions using `ccauth`.

24.5.1 Managing volumes via the GUI

Creating/Editing/Deleting volumes

1. Login to KVM@TACC using your Chameleon account.
2. Navigate to the Volumes overview under "Volumes > Volumes" in the sidebar. Select "Create Volume" to create a new volume. Enter a name and a size in Gigabytes. Under type, select either "ceph-hdd" or "ceph-ssd." The storage type "ceph-ssd" is backed by a smaller set of nodes with SSD storage. This type of volume will be relatively performant (the same as your instance's root partition), but only a small portion of capacity is on the SSDs. The "ceph-hdd" type is backed by spinning disks, and you may experience slow performance doing random access. Click "Create Volume".

3. On the volume overview page, you can rename your volume via "Edit Volume". By selecting the action arrow, you can click "Extend Volume" to increase its size. By selecting "Change Volume Type" you can switch between "ceph-ssd" and "ceph-hdd".
4. To delete your volume, on the volume overview page you can select the action arrow click "Delete Volume".

Attach/Detach volumes

This guide assumes you have a running instance (see *Launching instances*).

After creating your volume, you can attach it to your instance by selecting the "Manage Attachments" action. On the "Manage Volumes Attachments" dialog, pick your instance from the dropdown and confirm by clicking "Attach Volume". On the Volume overview, you'll now see which device the volume will appear inside your instance.

In order to use the new block device, you'll need to parition, format, and mount it inside your instance. These instructions may vary depending on your operating system, but for more information see these links on how to partition and how to mount block volumes.

In the future, you will not need to partition and format the volume, and can just mount it after attaching.

24.5.2 Managing volumes via python-chi

See this Trovi artifact for how to manage volumes via python-chi.

CHAPTER

TWENTYFIVE

PACKAGING & SHARING EXPERIMENTS

Chameleon Trovi is a sharing portal that allows you to share digital research and education artifacts, such as images, complex appliances, packaged experiments, workshop tutorials, or class materials. Each research artifact is represented as a deposition (a remotely accessible folder) where a user can put Jupyter notebooks, links to images, orchestration templates, data, software, and other digital representations that together represent a focused contribution that can be run on Chameleon. Users can use these artifacts to recreate and rerun experiments or class exercises on a Jupyter Notebook within Chameleon. They can also create their own artifacts and publish them directly to Trovi from within *Chameleon's Jupyter server*.

Note

Trovi is now the sole home for Chameleon appliances as well as user experiments. All Chameleon-supported OS *images* and *Complex Appliance* heat templates are published and discovered on Trovi by filtering for the **appliance** tag — the legacy Appliance Catalog is deprecated and no longer used.

To get started, find the "Trovi" dropdown option under the "Experiment" section of chameleoncloud.org. Once you're on the Trovi homepage, you'll see a list of publicly available experiments and other digital artifacts. You can now browse those artifacts or upload your own.

Once your artifact is packaged and shared, you may also want non-Chameleon users to be able to reproduce it — see *Daypass: temporary access for reproduction* for how to grant reviewers and collaborators temporary access without a full Chameleon allocation.

25.1 Browsing & launching artifacts

Trovi allows you to browse artifacts, presented in a scrolling list format, and to search or filter to narrow down results. You can also see how many times people have downloaded and launched an artifact via the icons in the bottom left corner of each entry.

25.1.1 Searching & filtering artifacts

On the Trovi homepage, the **Browse Artifacts** section provides a search bar where you can type keywords to find relevant artifacts. The search supports a few operators to refine your results:

- Wrap a phrase in quotes (e.g. `"data science"`) to search for an exact match.
- Use `OR` to broaden the search across multiple terms.
- Prefix a word with `-` (e.g. `-jupyter`) to exclude it from results.

Beyond keyword search, you can narrow results further using the options on the right-hand side:

- **Tags**: filter by category, such as `education`, `experiment`, `reproducible research`, or `appliance`.

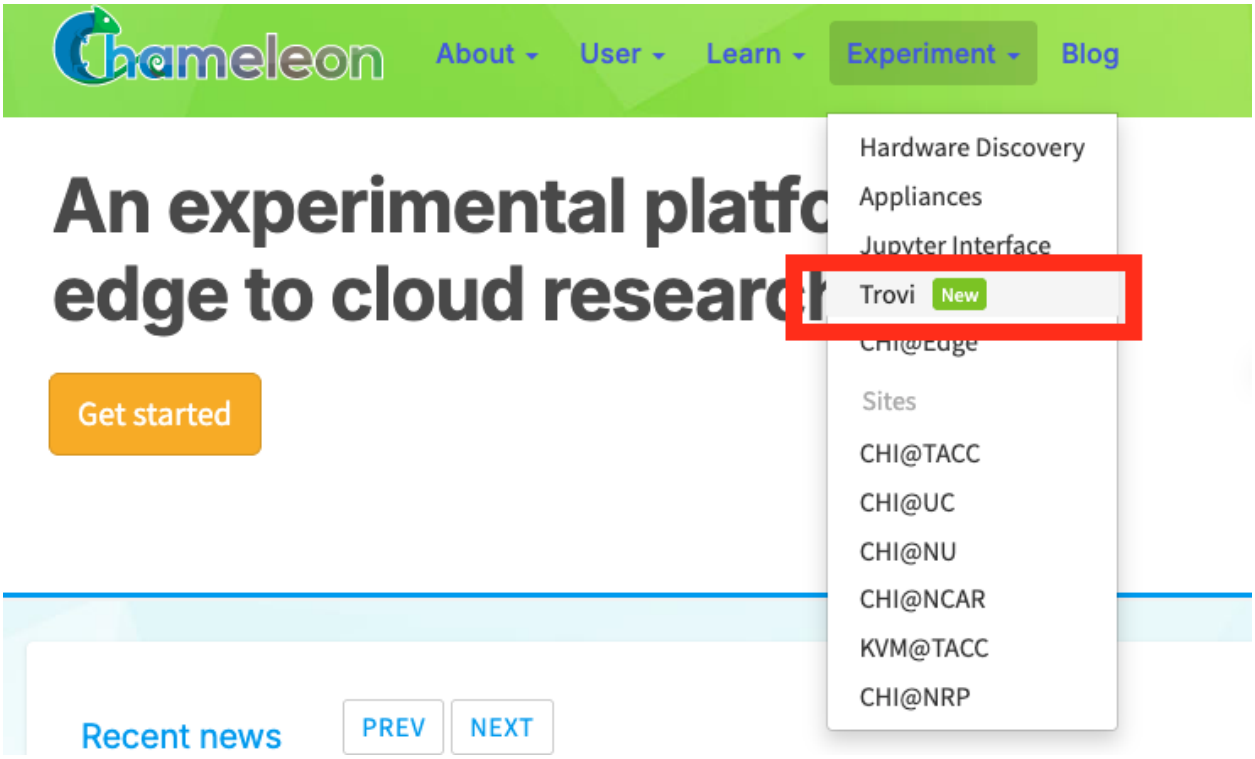


Fig. 1: The "Trovi" option under the "Experiment" section takes you to Trovi.

- **Badges**: filter by badge, such as `chameleon supported`, `reproducible`, or `educational`. Artifacts supported by the Chameleon team display a small Chameleon logo; you can contact the Help Desk if you are using one of these artifacts and encounter issues.
- **Filters**: narrow results to only your own artifacts, only public artifacts, collections, or artifacts with a DOI.

Results can also be sorted by relevance or other criteria.

Most artifacts on Trovi are user-uploaded, but we have some external artifacts marked with an **external artifact** label in the description that were brought in from outside sources rather than packaged directly on Trovi; see the Tips and Tricks post Bringing External Reproducibility Artifacts Into Trovi for details on how these are imported and launched.

> **Tip**
>
> Chameleon appliances — including Chameleon-supported OS images and heat templates — are published on Trovi and discoverable via the **appliance** tag. Whether you are looking for a Jupyter notebook or a bare-metal image, you can find either one via Trovi.

25.1.2 Viewing artifact details

Click on an artifact of interest from the search results to open its detail page. At the top you'll find the artifact's title and at-a-glance stats — total launches, unique users, users who launched on Jupyter, versions, and last updated date — followed by an **About** section describing generally what the artifact does, how it works, hardware requirements, expected runtime, and outputs. The right-hand sidebar includes a launch action (see below), an **Authors** list with institutional affiliations and contact emails, a **Versions** panel for browsing prior releases, a **Citation** section with ready-to-copy standard and BibTeX citations, and a link to the artifact's source repository (e.g., GitHub) if you want to explore the underlying files.

25.1.3 Launching an artifact

The most powerful feature available via Trovi is the ability to re-launch the available artifacts within Chameleon. Clicking "Launch with JupyterHub" will open a new Jupyter Notebook server with the artifacts downloaded (we support artifacts up to 500MB in total size, contact the Help Desk if you need more space). The animation below shows how easy it is:

Fig. 2: Clicking the "Launch with JupyterHub" button to import a Trovi artifact into your own Jupyter server.

25.2 Packaging shared artifacts

You can publish new artifacts to Trovi either from your primary Jupyter server or by editing a previously-shared artifact. In the latter case, you are effectively creating a new "forked" artifact owned by you.

When you've finished creating or making changes to an experiment, in the Jupyter interface, select the directory (not a single file) you wish to package. Then, click on the "Share" tab and select "Package as a new artifact". Your artifact is now packaged and uploaded to Chameleon file storage, and you'll be prompted to fill out descriptions about the artifact. Don't worry if you want to change this later—you will be able to *edit them on the Trovi portal or within Jupyter*.

Congratulations! Your artifact is now uploaded to Trovi—but to make it accessible to others you need to *adjust its sharing settings*.

25.2.1 Saving new versions

If you make changes to your artifact, you can submit an updated version. Within Jupyter, you navigate to the "Sharing" tab, but this time you click "Create new artifact version". The different versions are viewable on the Trovi portal after clicking on the artifact.

25.2.2 Editing artifacts

You can edit an artifact's metadata, including its title, description, and list of authors at any time via the Jupyter interface. To delete single artifact versions, click the "trashcan" icon next to it in the edit view. To delete the entire artifact, click the red "Delete All" button.

Note

Any artifacts published to *Zenodo* cannot be deleted.

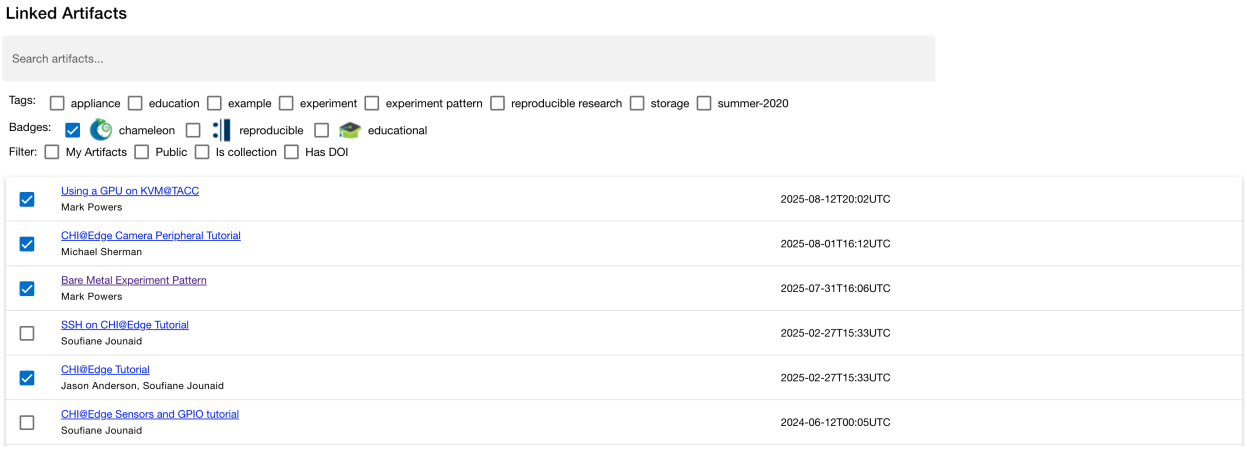
This edit view is also available from Trovi via the "Edit" button.

Creating a version from Git

Under an artifact's edit settings, you will also see a button for creating a new version from Git. This will allow you to enter a Git URL, the same one used to clone your repository, and also a reference (lease as HEAD for the latest commit). When a user launches your artifact, their notebook will checkout the specified commit.

Editing related artifacts

From the artifact edit page on the Trovi dashboard, you can also add links to other related artifacts on Trovi. This can be useful if you have multiple artifacts that are part of a related project, or if you'd like to showcase a collection of artifacts



on a broad topic, e.g., a series of assignments supporting a course on machine learning operations. All artifacts that you link to your artifact will appear under "Related Artifacts" after the artifact's description.

While editing an artifact, under "Linked Artifacts", you will see a table of all artifacts. Search or scroll to find the artifact you wish to link, and click the checkbox next to it. Below this table, you can rearrange the order of linked artifacts by dragging and dropping them. Click "Save Links" to save your changes.

25.2.3 Adjusting sharing settings

When you first upload your packaged artifact to Trovi, its visibility is set as private, meaning only you can see or launch it. There are multiple options to change the visibility of the artifact, and you have the option to decide how visible you want it to be.

1. **Publish with DOI**: this option allows you to *publish a version of your artifact to Zenodo* and receive a DOI, which you can use to cite your artifact in, e.g., an academic paper.
2. **Publish without DOI**: this option allows any Chameleon user to find and launch your artifact. It can be useful if you want to distribute the artifact widely but do not necessarily with to publish it to Zenodo and get a DOI for citation.
3. **Share via private link**: this option allows you to share the experiment to select people, like individual colleagues, advisors, or students. Anybody in possession of the link can view and launch any version of the artifact.

To make your artifact shareable, select it in Trovi, click "Share", and check the box before "Enable all users to find and share".

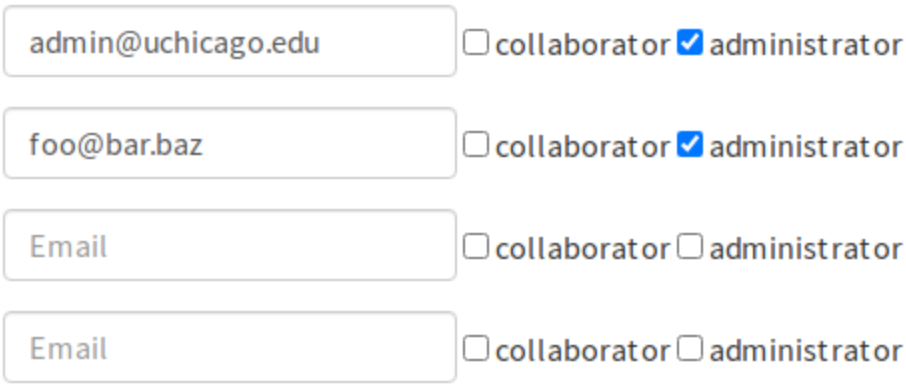
Assigning roles to other users

You can assign roles to other users which allow them to collaborate on your artifacts. There are two roles: **Collaborator** and **Administrator**.

Collaborators are allowed to edit artifact metadata, upload new versions, delete old versions, and share private artifacts.

Administrators have full control over the artifact, including assigning roles to other users.

Artifact owners cannot have their Adminstrator privileges removed.



Publishing to Zenodo

Attention

You can only request a DOI for artifacts uploaded via the Jupyter interface. You cannot request a DOI for an artifact version uploaded via git.

Trovi is intended for sharing work in progress with a limited group of "friends and family". However, once you complete your experiment package you may want to publish it so that you can reference it from your paper. To do that Chameleon supports integration with Zenodo, an open-access storage repository backed by CERN, for permanent artifact hosting. To share your artifact and store it on Zenodo, go to the "Share" page for the artifact. On the right-hand side you'll see a list of all versions you've saved. Pick the version you want to publish to Zenodo and check "Request DOI", then click "Save."

Important

Once published, **Zenodo artifacts cannot be deleted** and are additionally **publicly available**. Your artifact will appear in Trovi in the "Public" section, and any Chameleon user can access it, as can anybody on the Internet via Zenodo's own listing.

If you wish to make your artifact public but don't to publish it, use the "Publish without DOI" option. With this option it is possible to make the artifact private later on if you wish; this is not possible when publishing to Zenodo.

You can only request a DOI one time per artifact. If you want to update your experiment files and request a second DOI, you should instead create a new artifact.

This also creates a DOI, which you can easily include in your paper. The artifacts shared on Zenodo also appear on Trovi.

Tip

Publishing to Zenodo is one way to keep your work available long-term. For a full rundown of how long Chameleon resources and artifacts are retained by default, and other options for extending their lifespan, see the Tips and Tricks post Extending Your Research Artifacts' Lifespan.

25.2.4 Importing an artifact

Instead of creating an artifact inside Jupyterhub, you can package an existing Git repository into an artifact. When a user launches the artifact, the contents of the repository will be added to a Jupyter notebook.

To create an artifact, click "Import Artifact" on the sidebar of Trovi. You are first asked for the artifact's metadata. At the bottom of the form, there is a button for "Import from Git." After clicking this, you will need to enter a git remote URL, and choose which commit to tie the version to.

To update the artifact, you must create a *new version*. This ensures that a given version of your artifact always has the same contents.

> **Tip**
>
> You can also generate the artifact's metadata ahead of time from the command line using the `troviclient` tool, which produces a `trovi.json` file to commit alongside your repository. See the Tips and Tricks post Importing GitHub Repositories to Trovi: A Step-by-Step Guide for a walkthrough.

25.2.5 Exporting via git

If you wish to move your code and notebooks outside of your Jupyter notebook, one option is to export it into a git repository.

1. Click the "+" button on the top left of your notebook, and choose "Terminal".
2. Run the command `cd work`. If there is a specific directory you wish to export, you can `cd` again into it.
3. Follow the instructions to set up a repository per your git host. For GitHub see this document.
4. After the repository is set up, you should be able to commit and push with the git CLI.

25.3 Daypass: temporary access for reproduction

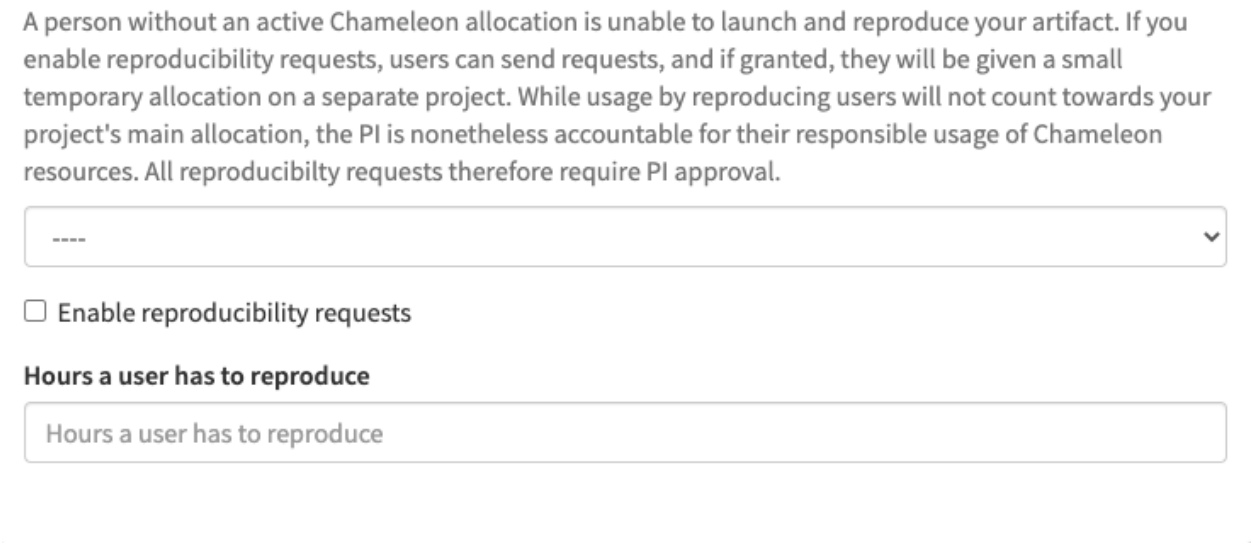
Normally, only Chameleon users with active allocations are able to launch and view Trovi artifacts. To allow anyone to launch an artifact, we also provide daypass. This allows for a non-Chameleon user to have access to Chameleon for a limited amount of time, using a small, separate allocation. People interested in reproducing your project will send requests to the managers of a project. If approved, the requesting user will receive an email invitation to join a reproducibility project. When they accept, they can use this project to run your artifact. After the specified time limit, they will be automatically removed from this project. Daypass can be enabled on an artifact-by-artifact basis in Trovi.

> **Tip**
>
> If you need to support a structured artifact evaluation (AE) process with multiple reviewers, separate author/reviewer projects, and per-reviewer compute budgets, rather than one-off reproduction by individuals, see the Tips and Tricks post Running Artifact Evaluations on Chameleon.

25.3.1 Enabling daypass

To enable Daypass, the owner of an artifact first must permit reproducibility requests. This can be revoked at any time, preventing future requests. Additionally, you must also give your artifact a value for "Hours a user has to reproduce." This value specifies how long a user will have access to Chameleon for. Consider how long it takes to run your experiment from start to finish as a lower bound for this value. The artifact owner must also assign their artifact to a project via the dropdown selector. As these requests are granting access to Chameleon resources, this is needed to tie granted requests to a PI.

These fields can be accessed by navigating to an artifact's detail page, and then selecting "Share." At the bottom of the share page, you will see the below forms, which are the project assignment, the enabling of reproducibility requests, and the hours to reproduce.



After these items are saved, a reproducibility allocation request is automatically made under your PI's name. Your artifact should now appear with a "Request Daypass" button below the "Launch" button. The "Launch" button will not appear for users that are not a member of an active Chameleon project.

25.3.2 Requesting a daypass

When another researcher wishes to reproduce your artifact (or when you wish to do so for another's artifact), selecting "Request Daypass" will display a form asking for a name, institution, and the reason for reproducing the artifact. For artifacts under your project/allocation, this gives you oversight and discretion as to who you grant access to, as ultimately you're responsible for usage under your allocation (including reproducibility allocations). For artifacts you wish to explore, it gives you the chance to reach out to the project owners and explain why you are interested in their work.

After submitting the form, the managers (and PI) of the project associated with the artifact will receive an email informing them of the request.

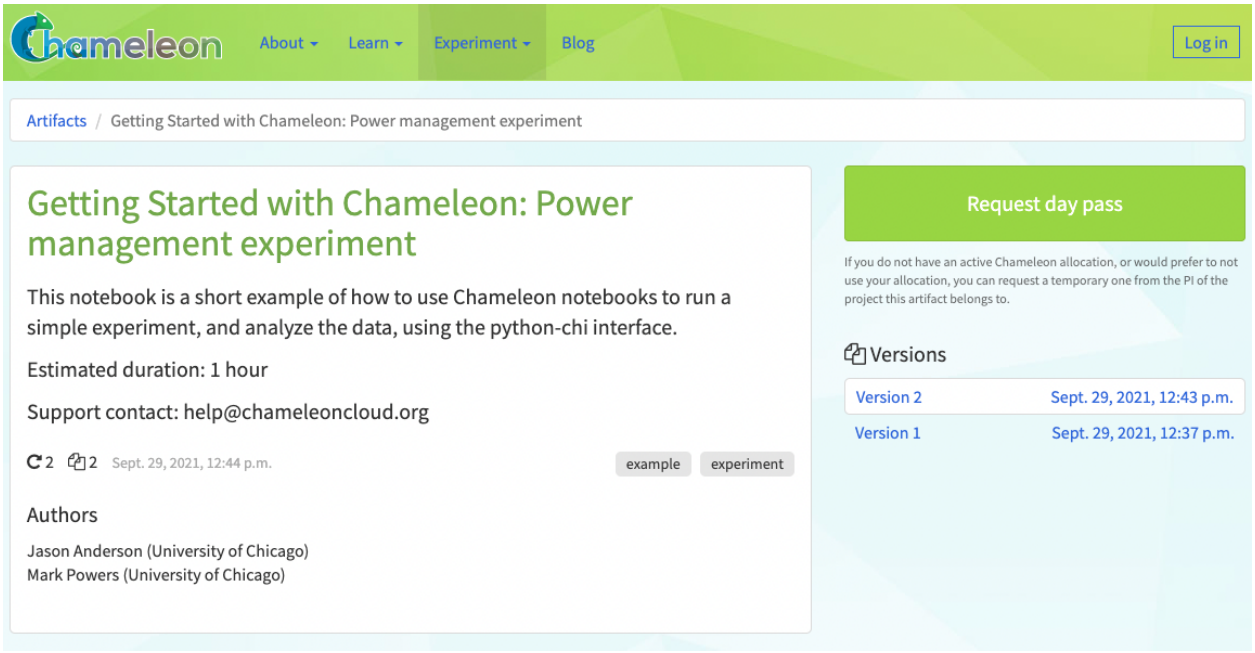
25.3.3 Reviewing a daypass request

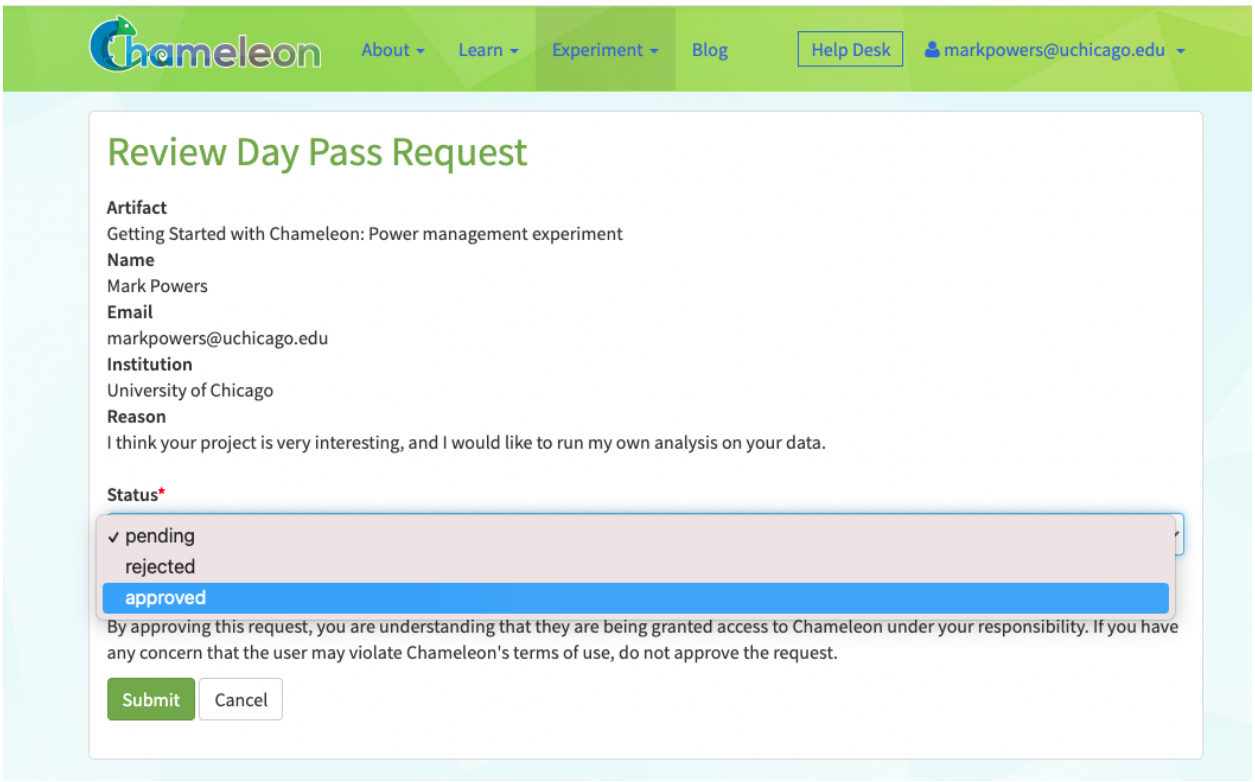
After receiving an email with the daypass request, PIs and project managers can navigate to the review page by clicking the link in the email. Here, they will see all of the details submitted with the request. A decision can be made by choosing "approved" or "rejected" in the selector, and then clicking submit.

After this decision is made, an email is sent to the requestor with the result. If the request is approved, an invitation is sent to the user.

25.3.4 Using an invitation

If your daypass request is approved, an email will be sent to you with an invite link. After clicking this link, you will be automatically added to the project. The email will also mention how long the invitation is for. When the invite is accepted, you will be taken to the project page for the reproducibility project. Note the ID of the project (CHI-XXXXX), which may be needed to configure an artifact.


Chameleon
About
Learn
Experiment
Blog
Log in
Artifacts / Getting Started with Chameleon: Power management experiment
Getting Started with Chameleon: Power management experiment
This notebook is a short example of how to use Chameleon notebooks to run a simple experiment, and analyze the data, using the python-chi interface.
Estimated duration: 1 hour
Support contact: help@chameleoncloud.org
2 2 Sept. 29, 2021, 12:44 p.m.
example
experiment
Authors
Jason Anderson (University of Chicago)
Mark Powers (University of Chicago)
Request day pass
If you do not have an active Chameleon allocation, or would prefer to not use your allocation, you can request a temporary one from the PI of the project this artifact belongs to.
Versions
Version 2 Sept. 29, 2021, 12:43 p.m.
Version 1 Sept. 29, 2021, 12:37 p.m.


Chameleon
About
Learn
Experiment
Blog
Help Desk
markpowers@uchicago.edu
Review Day Pass Request
Artifact
Getting Started with Chameleon: Power management experiment
Name
Mark Powers
Email
markpowers@uchicago.edu
Institution
University of Chicago
Reason
I think your project is very interesting, and I would like to run my own analysis on your data.
Status*
pending
rejected
approved
By approving this request, you are understanding that they are being granted access to Chameleon under your responsibility. If you have any concern that the user may violate Chameleon's terms of use, do not approve the request.
Submit
Cancel

Next, you can navigate back to the original artifact URL you were given. The "Launch" button can be used now to start running the artifact.

After the duration for the invite has passed, you will be automatically removed from the project.