Introduction

1 Welcome to Chameleon  
2 Getting started  
3 Sign in with federated identity  
4 PI eligibility  
5 Project management  
6 User profile  
7 Getting help  
8 Graphical User Interface (GUI)  
9 Command Line Interface (CLI)  
10 Jupyter interface  
11 Overview  
12 Resource discovery  
13 Reservations  
14 Bare metal instances  
15 Images  
16 Monitoring  
17 Complex appliances  
18 Object Store  
19 Networking  
20 FPGAs
1.1 What is Chameleon?

Chameleon is an NSF-funded testbed system for Computer Science experimentation. It is designed to be deeply reconfigurable, with a wide variety of capabilities for researching systems, networking, distributed and cluster computing and security. Chameleon’s features include:

- Bare metal access to hardware, via the cloud
- A wide variety of hardware types, including:
  - Infiniband
  - NVMe
  - GPUs/FPGAs
  - Low-power Xeon and ARM processors
- A powerful set of networking capabilities, including:
  - Isolated layer-2 networks
  - SDN/OpenFlow
  - Dedicated WAN layer-2 links

1.2 Using Chameleon

Chameleon is available to Computer Science researchers and students. To access the system, follow the instructions in Getting started. Find out more at https://www.chameleoncloud.org.
This guide will walk you through the initial steps of getting an account, joining a project and working with your first instance.

- **Step 1: Log in to Chameleon**
- **Step 2: Create or join a project**
- **Step 3: Start using Chameleon!**
  - The Chameleon dashboard
  - Reserving a node
  - Launching an instance
  - Associating an IP address
  - Accessing Your Instance

### 2.1 Step 1: Log in to Chameleon

Just click the “Log in” button situated in the top right corner of our main page – you probably won’t even need to create an account!

If your institution is a member of InCommon (most US research and education institutions are) – or if you have a Google account – you can log in with your institutional/Google credentials via the federated login. Otherwise, the log in process will guide you to create an account *(read more about logging into Chameleon via federated login)*.

On your first Chameleon login you will be asked to accept terms and conditions of use. Please, note that as part of those terms and conditions you are requested to acknowledge Chameleon in publications produced using the testbed: see our FAQ for information on how to reference Chameleon in your publications and the suggested acknowledgement text.
Once you log in, you will be able to edit your Chameleon profile, sign up for webinars, and participate in our community. However, to actually use the testbed you will first need to join or create a project (see below).

2.2 Step 2: Create or join a project

To get access to Chameleon resources, you will need to be associated with a project that is assigned a resource allocation.

If you want to join an existing Chameleon project, you will need to ask the PI of the project to add your username. You can find your username in your Chameleon profile—it is also displayed in the top-right corner when you are logged in.

If you want to create a project, you will first either need to obtain a PI status or work with somebody who has PI status. To determine if you can obtain PI status, please see a list of PI eligibility criteria. If you do not meet these criteria (e.g., students generally do not), you will need to ask your advisor or other scientist supervising your research to create the project for you. You can request PI status by checking a box in your Chameleon profile. Chameleon PI status requests are typically reviewed within one business day.

Once you have PI status, you may apply for a new project with an initial allocation. A project application typically consists of a short description of your intended research and takes one business day to process. Once your project has been approved, you will be able to utilize the testbed sites.

2.3 Step 3: Start using Chameleon!

Congratulations, you are now ready to launch your first instance! Instances are much like what you may expect to find in a virtual machine, except here the instances are on bare metal nodes—the core feature of Chameleon. A bare metal node is a whole physical server that you have exclusive access to. An instance is going to be a bare metal node that has been launched with an operating system image. Follow these steps to make a reservation for a node, launch an instance and log in to it.

Note: Chameleon also offers a multi-tenant, virtualized cloud, with fewer functionalities and a smaller scale. See KVM for more details.

2.3.1 The Chameleon dashboard

Chameleon resources are available at multiple sites, e.g., CHI@TACC and CHI@UC. When you access one of the sites, you are first taken to a dashboard, which shows a summary of your project’s current resource usage and get quick access to each of the sites. The dashboard looks something like this:

2.3.2 Reserving a node

First, we need to reserve a node for our use. Chameleon provides bare metal access to nodes. When you create a reservation for one or more nodes, only you and other users on your project will be able to use those nodes for the time specified. We will create a single day reservation for a compute node, which are the most common types of nodes available on Chameleon.

1. In the sidebar, click Reservations, then click Leases
2. Click on the + Create Lease button in the toolbar
3. Type my_first_lease for the lease name
4. Find the Resource Properties section. In the dropdown below node_type, select compute_haswell
5. Click the Create button

The reservation will start shortly, at which point you can launch an instance on a bare metal node.

**Note:** You have created an “on demand” reservation. When you do not specify a start date or time in the future, the reservation will start as soon as possible and will last one day.

**Important:** Do not attempt to stack reservations to circumvent the 7-day lease limitation. Your leases may be deleted. Please refer to our best practices if you require a longer reservation.

### 2.3.3 Launching an instance

Once the reservation starts, you can launch a bare metal instance on the node that has been leased to you.

1. In the sidebar, click Compute, then click Instances
2. Click on the Launch Instance button in the toolbar and the Launch Instance wizard will load
3. Type my_first_instance for the instance name and select your my_first_lease reservation
4. Click Source in sidebar. Then, find CC-CentOS8 in the image list and click the Up arrow to select it.
5. Click Keypair in sidebar. Click the + Create Key Pair button and enter mychameleonkey for the key name. This will automatically start a download for a file named mychameleonkey.pem. This is your private key pair that you will use to access your instance.
6. Click the Launch Instance button.
Create Lease

Lease Name
my_first_lease

Description:
Create a new lease with the provided values.
Leave date and time values blank to start a lease immediately (on-demand).
For specific node reservations, you can find the node UUID using Resource Discovery on the user portal.

Fig. 2: The Create Lease dialog - be sure to select compute(haswell) in the dropdown below node_type
2.3. Step 3: Start using Chameleon!

Fig. 3: Enter an instance name and select your reservation

Fig. 4: Select the CC-CentOS8 image
Chameleon Cloud Documentation, Release 2.0a

Congratulations, you have launched an instance on a bare metal node!

2.3.4 Associating an IP address

Your instance may take approximately ten minutes to launch. The launch process includes powering up, loading the operating system over the network, and booting up for the first time on a rack located either at the University of Chicago or the Texas Advanced Computing Center, depending on where you chose to launch your instance. 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.

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

2. If you have a 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 complete the process of associating the public IP to your instance.

![Manage Floating IP Associations](image)

**Fig. 6:** Here you can assign a floating IP address

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.

![Allocate Floating IP](image)

**Fig. 7:** This dialog allows you to allocate an IP address from Chameleon’s public IP pool

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.

---

2.3. **Step 3: Start using Chameleon!**
2.3.5 Accessing Your Instance

Once your instance has launched with an associated floating IP address, it can be accessed via SSH using the private key that you downloaded during the Launching an Instance step.

**Note:** The following instructions assume that you are using a macOS or Linux terminal equivalent. You may view our YouTube video on how to login via SSH on Windows.

To log in to your instance, follow these steps:

1. Open a terminal window and navigate to where you downloaded the `mychameleonkey.pem` file. Change the permissions on the file to user read/write only:

   ```bash
   chmod 600 mychameleonkey.pem
   ```

2. Add the key to your current SSH identity:

   ```bash
   ssh-add mychameleonkey.pem
   ```

3. Log in to your Chameleon instance via SSH using the `cc` user account and your floating IP address. If your floating IP address was `129.114.108.102`, you would use the command:

   ```bash
   ssh cc@129.114.108.102
   ```

   **Note:** Change the IP address in this command to match your instance’s floating IP address!
CHAPTER 3

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. 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. In addition, Chameleon also federates with the TAS entity.

3.1 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, KVM@TACC) or the Jupyter environment, just click their item in the “Experiment” dropdown on chameleoncloud.org. The login process is triggered automatically.

**Important:** You must be part of project with an active allocation to use the testbed sites! Refer to our “getting started” guide for more info.

**Note:** 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 a few options for how to authenticate. The options are:

- **Sign in via federated identity:** this option allows you to re-use your existing institution, research lab, or university credentials to log in to Chameleon. It requires your host institution to participate in the InCommon federation.
Chapter 3. Sign in with federated identity
• **Google**: this option allows you to sign in with any Google account.
• **ORCiD**: you can also sign in with a valid ORCiD account.
• **TAS**: sign in via the TAS entity.
• For the time being, if you are an existing user, you can also leave SSO and go back to the old sign in page, where you sign in with your Chameleon username and password.

![Fig. 1: The Single Sign On (SSO) portal login page.](image)

### 3.2 Legacy users (created before November 2020)

Users who signed up for a Chameleon account prior to November 2020 must migrate their account to use federated identity. Refer to the following step-by-step guide for more details.
3.2.1 Migrating to 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 institutional credentials to log into Chameleon—there is no need to create a new account. To leverage its many benefits, Chameleon is migrating to federated identity and will eventually drop support for signing in via legacy Chameleon accounts. Existing Chameleon users can link their existing Chameleon account to a federated identity account so that they preserve their project memberships, disk images, volumes, SSH key pairs, and other data saved over time.

To ease the transition, during the migration period you can log in via your old Chameleon account as well as the federated account. Existing users however are highly encouraged to migrate to a federated account as soon as possible. Read more about federated identity, how it has been implemented, as well as the migration schedule in our FAQ.

**Important:** Existing Chameleon users should go through the following account linking process **before** attempting to sign in with federated identity. Signing in before linking accounts can lead to you accidentally creating a new Chameleon account, if we cannot match the email address on your federated account to the one you used when registering for Chameleon.

**Step-by-step instructions**

1. **Sign in** to the Chameleon identity management interface. You will see a screen that prompts you for your existing Chameleon username and password.

2. Once authenticated, you will be taken to a page showing all the identities (credentials) linked to your Chameleon user account. **Click the “Add” button** next to the entry for the Globus identity to initiate another login via federated identity. You can pick whichever login method you choose; your federated identity need not be tied to the same email address or username as your existing Chameleon account if you wish.

   Once added, your linked federated identity can be used to log in to your existing Chameleon account in the future, and you no longer need to use your Chameleon username and password to login to any part of the Chameleon infrastructure.

3. If you try to log in to any of the testbed sites with your linked federated identity, you may notice that some of your data is missing, namely:

   - disk images, snapshots and volumes you or other project members have created in the past,
   - active or pending leases you made,
   - SSH keypairs you previously associated with your account, and
   - active server instances launched recently

   Migrating to federated identity effectively creates a new account for you in the system, so in order to retain access to some of your saved data, a short data migration is necessary.

   **Go to the migration page** in the Chameleon user portal to start this process (note: ensure you now log in to the user portal using federated login, and not the old sign-in page!)

   You should see a page that looks like the following.

   **Important:** Not all data can be migrated to your new account. Leases and active server instances will remain on your old account. If you wish to access these, you can log in to the testbed site using the old sign in method.

4. **Trigger a migration of your user** to copy over any SSH keypairs you previously associated with your account.
3.2. Legacy users (created before November 2020)

Federated Identities

- **Globus Auth**: 
- **TACC**: jason_a
5. **Trigger migrations of projects** you are a member of to associate any disk images, snapshots and volumes to your new account. This only needs to be done once by any member of the project, but can also be re-run in the event that you or other project members create disk images/snapshots under their old account.

6. You should now have access to your old data via your new account linked to your federated identity!

### Using the CLI

If you use the *command line interface* when interacting with Chameleon (or use another tool that interfaces directly with Chameleon’s APIs), you will no longer be able to authenticate with a username and password. You should re-download your *RC file* and use it when invoking the CLI, as it will have new authentication parameters compatible with your account pre-filled.

You can also look into generating an *application credential* for your command line client or app, which may be simpler.
PI eligibility

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 or faculty members in those institutions.

- **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.
Project management tasks, such as adding users to your project or requesting a renewal, is performed through the portal at https://chameleoncloud.org. After you have registered and verified your email address, you may login to the portal. Once logged in, you should be at Dashboard page automatically. If not, you can access your Dashboard via the dropdown list on top right of the screen.

5.1 Dashboard

The Dashboard's main page consists of two control panels - the Active Projects control panel and the Open Tickets panel.

![Fig. 1: The project dashboard](image-url)
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.

### 5.2 Projects

The Dashboard’s *Projects Page* allows you to manage your current projects.

![Fig. 2: Project list](image)

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

#### 5.2.1 Creating a Project

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

#### 5.2.2 Service Units

One Service Unit (SU) is equivalent to one hour of usage of one allocatable resource (physical hosts, network segments, or floating IPs). For example, a reservation for 5 Haswell compute nodes for 8 hours would use 40 SUs. However, for certain types of resources, more SUs will be charged. For more details about allocation charges, please see [here](#).
To find out more about applying for projects please visit our FAQ.

Title
Research into how...

Abstract (~200 words)
An application for a project has to include a description of the research or education project to be performed using the testbed and the type of resources needed. It should address the following questions: What are the research challenges or educational objectives of the project? How are they relevant to cloud computing research? Why are they important? What types of experiments or educational activities will be carried out? Please, make sure that the abstract is self-contained; eventually it may be published on the Chameleon website.

We propose to...

Resource Justification*
5.2.3 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.

![Project Details](image)

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

5.2.4 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 link next to your project in the Actions column of this section. 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!

5.2.5 Manage Publications

To add publications to a project, click the Add Publications button in the Project Details page. Please enter the publications in BibTex format. All regular BibTex publication types are supported. If you can provide a link, please enter as note or howpublished using the url package.
To manage the publications you have entered, use the *Publications Dashboard*.

![Publications dashboard](image)

Fig. 5: Publications dashboard

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

### 5.2.6 Manage User Roles (PI Delegate)

To manage user roles of a *Project*, use the *Project Members* section in the *Project Details*. The table below shows the types of roles and their privileges.

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PI</td>
<td>Each project has only one PI. PI can manage roles of the project members.</td>
</tr>
<tr>
<td>Manager</td>
<td>Each project can have multiple Managers. Managers can manage the membership of the project.</td>
</tr>
<tr>
<td>Member</td>
<td>Members can only view the list of the project members.</td>
</tr>
</tbody>
</table>

To change the role of a project member, click the *gear* button at the end of the user row, and then click *Change Role*. It will enable the role dropdown. Choose a new role and click the *Submit* button to apply the new role to the user, or use the *Cancel* button to cancel the action.

![Manage Role of a User](image)

Fig. 6: Manage Role of a User

### 5.2.7 Manage Users

To manage users of a *Project*, use the *Project Members* section in the *Project Details*. 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 a *PI eligibility*. 
You may add a user to your project by filling out their user name or email address and clicking the Add user button. 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.

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 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.
The Profile page in the Dashboard allows you to manage your biographical information and membership to any Chameleon mailing lists.

![Profile Page Screenshot]

**Fig. 1: The Profile page**

To edit your biographical information, click the **Edit Profile** button.


7.1 Mailing lists

You can communicate with other Chameleon users by sending email to users@chameleoncloud.org. We also use this mailing list to communicate minor announcements or provide early access to some new hardware or features. All Chameleon users are registered by default: we recommend that you stay registered. However, if you really want to opt-out, you can do so via your user profile.

7.2 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.

7.3 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.

Note: An alternative way of asking for help is sending an email to the Chameleon users mailing list, especially when the Help Desk is down or you think it’s something worth sharing with all Chameleon users. A system administrator will reply to your email and, if necessary, create a ticket for you.
Fig. 1: The Outages announcement page

Fig. 2: The Help Desk page
7.4 Webinars

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

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. There are two Chameleon resource sites, each with its own URL (though it is possible to easily switch from one to other, see Project and Site Menu).

- The Texas Advanced Computing Center resources (CHI@TACC) are available at:
  https://chi.tacc.chameleoncloud.org

- The University of Chicago resources (CHI@UC) are available at:
  https://chi.uc.chameleoncloud.org

Chameleon also hosts an OpenStack KVM implementation 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 features available on the GUI for the bare metal sites at the Texas Advanced Computing Center (CHI@TACC) and the University of Chicago (CHI@UC). For information about OpenStack KVM, please see KVM.

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

**Attention:** Each Chameleon testbed sites—CHI@TACC, CHI@UC, and KVM@TACC—are independent, so snapshots, keypairs, Swift containers, Gnocchi metrics 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.
8.2 GUI Features

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

![Fig. 1: The Chameleon GUI](image)

8.2.1 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.

8.2.2 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.

8.2.3 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.
Fig. 2: Switching between projects
Fig. 3: The user dropdown menu

Fig. 4: User settings
8.2.4 Help

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

8.2.5 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*.

8.2.6 Themes

You may change the GUI theme by selecting the provided menu items.

8.2.7 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.

8.3 Navigating the GUI

The navigation sidebar allows you to access different sections.

8.4 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.

8.5 Compute

Use *Compute* section for reserving, configuring and managing your instances.

8.5.1 Overview

The Overview page provides a graphical summary of your project’s current resource usage.
Chapter 8. Graphical User Interface (GUI)
API Access

Displaying 17 items

<table>
<thead>
<tr>
<th>Service</th>
<th>Service Endpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baremetal</td>
<td><a href="https://chi.uc.chameleoncloud.org:6385">https://chi.uc.chameleoncloud.org:6385</a></td>
</tr>
<tr>
<td>Baremetal Introspection</td>
<td>-</td>
</tr>
<tr>
<td>Cep</td>
<td><a href="https://chi.uc.chameleoncloud.org:8910">https://chi.uc.chameleoncloud.org:8910</a></td>
</tr>
<tr>
<td>Cloudformation</td>
<td><a href="https://chi.uc.chameleoncloud.org:8000/v1">https://chi.uc.chameleoncloud.org:8000/v1</a></td>
</tr>
<tr>
<td>Compute</td>
<td><a href="https://chi.uc.chameleoncloud.org:8774/v2.1">https://chi.uc.chameleoncloud.org:8774/v2.1</a></td>
</tr>
<tr>
<td>Compute_Legacy</td>
<td><a href="https://chi.uc.chameleoncloud.org:8774/v2/975c0a94b784483a885f4503f70af655">https://chi.uc.chameleoncloud.org:8774/v2/975c0a94b784483a885f4503f70af655</a></td>
</tr>
</tbody>
</table>

Fig. 5: The API Access page
Note: At the bare metal sites, you may launch as many instances as you like, but bounded by the project Service Unit allocation. However, at the OpenStack KVM site, your project is limited to a certain number of virtual machines. By default, each project is allowed to allocate 50 Floating IP addresses and use 10 Security Groups. You may request additional resources by submitting a ticket on the Help Desk.

8.5.2 Instances

The Instances page allows you to work with your instances. You may launch, terminate, monitor, or reboot an instance. Clicking on the dropdown list in Action column to see what you are eligible to do to your instances.

8.5.3 Images

The Images page allows you to view, upload and edit the images. You may also use this page to launch instance using selected images.

Note: You can only edit the images you own.
8.5.4 Key Pairs

The Key Pairs page allows you to create, import and manage SSH key pairs associated with your user account.

**Note:** Chameleon only stores the public key for each SSH key pair. **Do not** upload your private key to the portal! Private keys look like this:

To delete a SSH key pair, click on the *Delete Key Pair* button in the *Action* column. You may delete multiple key pairs by selecting them via the checkbox and clicking the *Delete Key Pairs* button.

**Creating a Key Pair**

To create a key pair, click the *Create Key Pair* button. In the prompted dialog, provide a name for your new key pair and then click the *Create Key Pair* button.

![Fig. 6: Specifying a key pair name](image)
A .pem file that contains the Private Key should be automatically downloaded. In addition, the Public Key associated with the Private Key should be saved automatically to Chameleon. Clicking on the Regenerate and download Key Pair button will generate a new Public/Private Key Pair and initiate a new download of the Private Key.

**Tip:** Save the Private Key to a location you will remember at your local file system. Your home directory is recommended for macOS and Linux systems.

### Importing a Key Pair

Alternatively, you may import a key pair that you have generated on your computer. Clicking the Import Key Pair button to prompt the dialog. Then, provide a name for your imported key pair and paste the Public Key.

**Tip:** The prompted dialog contains the instructions on how to generate a key pair using the Linux/macOS command.

![Fig. 7: Importing a public key](image)

**Tip:** 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.
8.6 Network

The Network section allows you to work with virtual network resources, such as configuring routers and virtual networks. For more information, please see Networking.

8.6.1 Network Topology

The Network Topology page displays your current virtual network topology in either the Topology or Graph formats. You may also use this section to directly launch instances, create networks or create routers.

![Fig. 8: The Network Topology page](image)

8.6.2 Networks

The Networks page lists all the Virtual Networks of the selected project. You may use this section to create, delete and modify Virtual Networks. Clicking on the dropdown list (if shown) in Action column to see what you are eligible to do to your virtual networks.

![Fig. 9: The Networks page](image)

8.6.3 Routers

Same as the Networks page, the Routers page allows you to work on the Routers of the selected project.
8.6.4 Security Groups

Use the Security Groups page to create, delete, and modify the Security Groups of the selected project.

Attention: Chameleon bare metal sites—CHI@TACC and CHI@UC—do not support security groups, i.e., all ports are open to the public.

8.6.5 Floating IPs

The Floating IPs page allows you to work with the Floating IP addresses allocated for the selected project, including associating with instances and releasing back to the pool. Clicking on the dropdown list (if shown) in Action column to see what you are eligible to do to your Floating IPs.

Releasing Floating IP Addresses

Important: The Chameleon Floating IP address pool is a shared and finite resource. Please be responsible and release the Floating IP addresses that are not used, so other Chameleon users and projects can use them!
Fig. 12: The Floating IPs page

Fig. 13: Releasing a Floating IP address
8.7 Orchestration

The Orchestration section allows you to work with the *Chameleon's Complex Appliances*.

8.7.1 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. 14: The Stacks page](image)

**Tip:** After launching a stack, all the instances launched with the stack can be viewed at *Compute - Instances* section as well.

**Note:** When you terminate a stack, all instances launched with the stack will be terminated.

8.7.2 Resource Types

The Resource Types page lists the currently available Orchestration Resource Types of Chameleon. You may click on the resource types to get details. The Orchestration Resource Types are used when writing *OpenStack Heat Orchestration Template*. For more information about *OpenStack Heat*, please see the *OpenStack Heat* documentation.

![Fig. 15: The Resource Types page](image)
8.7.3 Template Versions

The Template Versions are also used when writing *OpenStack Heat Orchestration Template*. Clicking on the version to get supported features of the specific version.

![Template Versions page](image)

Fig. 16: The Template Versions page

8.8 Object Store

The *Containers* section under *Object Store* gives an easy access to your Chameleon object/blob store. You may create, delete, upload objects to or remove objects from containers via this page. For more information about Chameleon Object Store, please see *Object Store*.

![Containers page](image)

Fig. 17: The Containers page

8.9 Reservations

The Reservations section allows you to manage your leases of the selected project, including creating and deleting leases. For more information, see *Reservations*.

*Tip:* Check *Lease Calendar*, so you can schedule your experiments efficiently.
8.10 Identity

The Project section under Identity allows you to check what projects you belong to. You can set your default project by clicking the *Set as Active Project* button in the *Actions* column.

---

Fig. 18: The Leases page

---

Fig. 19: The Projects page
9.1 Introduction

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.

**Attention:** Some of the Chameleon features are only accessible via the CLI, such as the Gnocchi metrics and the advanced networking features.

**Note:** 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.

9.2 Installing the CLI

9.2.1 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.

9.2.2 OpenStack Client Installation

1. Install the CLI by typing `pip install python-openstackclient` 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`.

4. There are some clients with new features or bugfixes not yet in the upstream release branches, notably the Blazar CLI client. If you want to make reservations via the CLI, you should install that here:

   ```sh
   pip install git+https://github.com/chameleoncloud/blazar@chameleoncloud/train
   ```

### 9.3 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. There are two ways of doing this.

#### 9.3.1 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.

![Change Password](image)

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 (<https://www.lastpass.com/password-manager>), or 1Password to assist.

#### 9.3.2 Creating an application credential

You can also generate **application credentials**, which act as dedicated one-off passwords that are authorized with the same permissions as your user account, 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.

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.

9.3. CLI authentication
### 9.4 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*.

**Hint:** If you use the Chameleon supported (CC) images, you’ll find an *openrc* file with a service token in the home directory for the *cc* user. The file will be auto-sourced when you login, so you can use the *openstack* and the *swift* CLI directly, as well as the *cc-snapshot* and the *cc-cloudfuse* tools.

1. Log in to the GUI at CHI@TACC or CHI@UC.

   **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*.

   ![Fig. 2: The Project Dropdown](image)

3. Download *OpenStack RC Script* using *User Menu* by clicking on *Openstack RC File v3*.

4. Run the following command in the terminal:

   ```bash
   source <path/to/openstack_rc_file>
   ```
Fig. 3: The OpenStack RC File v3 link in the User Dropdown
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.

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:

   ```bash
   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.

**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, please 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.
Fig. 4: System Properties Window of Windows System
9.5 Using the CLI

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`.

9.5.1 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.

9.5.2 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.
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 Jupyter-Hub 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.

### 10.1 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, please 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.
10.2 Working with Notebooks

Open the “Welcome.ipynb” Notebook to see some examples of how to interface with the Chameleon testbed from within a Notebook. All Notebook servers come with OpenStack python clients installed as well as the python-chi Chameleon testbed helper library. Other python modules you may want to use in your Notebook can be installed via the Console interface.

10.3 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.

10.4 Advanced topics

10.4.1 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
Welcome to Jupyter on Chameleon!

Getting started

Before you do anything, you should first set up a few project variables. This is so the Jupyter Notebook knows which project to operate on. If you are only a member of one project, you can skip this step, as the project will be selected for you by default. You can also set the site you want to use via the `region_name` setting - this defaults to CH1@UC.

```python
import chi

chi.set('project_name', '')  # Replace with your project name
chi.set('region_name', 'CH1@UC')  # Optional, defaults to 'CH1@UC'
```

Launching a baremetal server

Starting a new baremetal server is easy to do.

```python
from chi.lease import Lease

# Create a lease for 1 node
lease = Lease(node_type='compute_baswell')
lease.wait()  # Ensure lease has started

# Launch 1 server
server = lease.create_server(image='CC-CentOS7')
server.wait()  # Ensure server has started
```
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.

**Using the Appliance Catalog**

The Chameleon **Appliance Catalog** provides a JupyterHub appliance that is functionally equivalent to the shared Jupyter environment. The appliance will allow you to reserve a Chameleon bare metal node and provision it with the JupyterHub application, along with a **Floating IP Address** that allows access over the public Internet. Any Jupyter Notebook servers managed by this multi-user environment will have access to the underlying resources on whatever node you have reserved, removing the limits around CPU and memory usage.

**Using Trovi**

_Trovi_ also has a JupyterHub artifact you can instantiate on Chameleon. There is no material difference between this method and the Appliance Catalog method and it can serve as a nice introduction to the Sharing Portal if you are not already familiar with it. With this method, you actually provision your own JupyterHub server _via the shared default Chameleon JupyterHub_; it’s JupyterHub all the way down!

### 10.4.2 Collaboration Strategies

It is often desirable to share your in-progress Notebooks with peers or supervisors for feedback, perhaps before publishing in _Trovi sharing portal_. 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.
Overview

This section provides in-depth knowledge for utilizing Chameleon’s advanced features.

- **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.
- **Monitoring**: Collect, manage and view experimental data from Chameleon Instances.
- **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.
- **Networking**: Create Isolated virtual networks within Chameleon.
- **FPGAs**: Configure and work with FPGA nodes.
- **KVM**: Use non-bare metal virtual machine resources in Chameleon’s OpenStack implementation.
12.1 Introduction

Chameleon supports fine-grained resource discovery for experimentation, which means that you can identify a specific node, view the node’s hardware maintenance history and reserve it for repeated use.

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 web interface or directly via REST APIs.

Note: Some resource discovery features are available through the Chameleon Portal, while others are available only through the REST APIs.

12.2 The Hardware Catalog on the Chameleon Portal

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

12.2.1 Availability

The CHI@TACC and CHI@UC buttons in the Availability section of the Resource Browser allow you to open the Lease Calendars at the Chameleon sites. You must login using your Chameleon account to view these lease calendars.

12.2.2 Chameleon Resource Browser

The Chameleon Resource Browser allows you to filter Chameleon resources by node type and view details of each node.
Fig. 1: Resource availability links to the lease calendars

Fig. 2: The Chameleon Resource Browser
You may filter for specific node types by selecting the checkboxes that match your filter criteria or by clicking the buttons such as *Compute* and *Infiniband Support*. The numbers printed next to the node types indicate the total number of nodes of that certain type. After you have selected filter criteria, you can click the *View* button to see details of individual nodes that match your filtering criteria.

![Node details](image)

**Fig. 3: Node details**

**Tip:** To get more precise characteristics of the selected node, search the node at [Intel’s CPU database](https://www.intel.com/content/www/us/en/processors.html).

**Note:** All the nodes in Chameleon is identified by their *UUIDs*. You will need the *UUID* of a node for making reservations and identifying metrics collected from the node using Gnocchi. 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.

### 12.2.3 Generating a Reservation Script

The Chameleon Portal does not support a direct reservation from the *Hardware* page. However, you may generate a script for reserving the selected nodes by clicking on the *Reserve* button and use the auto-generated script later for the reservation.

After the form is submitted by clicking the *Generate Script* button, a new dialog that contains the auto-generated command line will show.

For node reservation using auto-generated command, please see *Provisioning and Managing Resources Using the CLI*.

---

**12.2. The Hardware Catalog on the Chameleon Portal**
Fig. 4: Generating a reservation script
12.3 Using the REST APIs for Resource Discovery

The API is designed for users who want to programmatically discover Chameleon resources. It uses a REST architecture on top of the HTTP protocol. As a consequence, any HTTP client can be used to query the API: command-line tools (cURL), browsers, and the numerous HTTP libraries available in your favorite programming language.

It also implements the concept of “Hypermedia as the Engine of Application State” (HATEOAS), by specifying a set of hyperlinks in all responses returned by the API, which allow a user agent to discover at runtime the set of available resources as well as their semantics and content types, and transition from one resource to another.

12.3.1 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):
Chameleon Cloud Documentation, Release 2.0a

$ sudo yum install curl

Windows
Download and install the cURL package from the website.

12.3.2 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/1.1 200 OK
Server: nginx/1.6.2
Date: Thu, 19 Apr 2018 14:34:01 GMT
Content-Type: application/vnd.grid5000.item+json; charset=utf-8
Content-Length: 757
Connection: keep-alive
Allow: GET
Vary: accept
Last-Modified: Wed, 14 Mar 2018 15:05:58 GMT
ETag: "cc990a75afbc3aed5979c5cad2358b14"
Cache-Control: max-age=60, public, must-revalidate=true, proxy-revalidate=true, s-maxage=60
X-Info: Use `?pretty=yes` or add the HTTP header `X-Rack-PrettyJSON: yes` if you want pretty output.
X-UA-Compatible: IE=Edge,chrome=1
X-Runtime: 0.034541

{"type":"grid","uid":"chameleoncloud","version":"ee0253a05223dd0f5b88df7f78f8988e67f7b039","release":"3.5.7","timestamp":1524148441,"links": [{"rel":"sites","href":"/sites","type":"application/vnd.grid5000.collection+json"},{"rel":"self","href":"/","type":"application/vnd.grid5000.item+json"},{"rel":"parent","href":"/","type":"application/vnd.grid5000.item+json"},{"rel":"version","href":"/versions/ee0253a05223dd0f5b88df7f78f8988e67f7b039","type":"application/vnd.grid5000.collection+json"},{"rel":"users","href":"/users","type":"application/vnd.grid5000.collection+json"},{"rel":"notifications","href":"/notifications","type":"application/vnd.grid5000.collection+json"}]
```

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

Tip: By default the response body is not displayed in a pretty format. You must add the pretty query parameter to the end of the URI if you want the API to display it in a prettier way. `curl -i https://api.chameleoncloud.org/?pretty`
**Attention:** Do not use the pretty query parameter in your scripts, since it requires a bit more processing power to generate.

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.

```bash
curl https://api.chameleoncloud.org/sites?pretty
```

The response should look like:

```json
{
  "total": 2,
  "offset": 0,
  "items": [
    {
      "description": "Texas Advanced Computing Center",
      "email_contact": "help@chameleoncloud.org",
      "latitude": 30.390223,
      "location": "Austin, Texas, USA",
      "longitude": -97.72563,
      "name": "TACC",
      "security_contact": "help@chameleoncloud.org",
      "sys_admin_contact": "help@chameleoncloud.org",
      "type": "site",
      "uid": "tacc",
      "user_support_contact": "help@chameleoncloud.org",
      "web": "https://www.chameleoncloud.org",
      "version": "ee0253a05223dd0f5b88df7f78f988e67f7b039",
      "links": [
        {
          "rel": "clusters",
          "href": "/sites/tacc/clusters",
          "type": "application/vnd.grid5000.collection+json"
        },
        {
          "rel": "self",
          "type": "application/vnd.grid5000.item+json",
          "href": "/sites/tacc"
        },
        {
          "rel": "parent",
          "type": "application/vnd.grid5000.item+json",
          "href": "/"
        },
        {
          "rel": "version",
          "type": "application/vnd.grid5000.item+json",
          "href": "/sites/tacc/versions/ee0253a05223dd0f5b88df7f78f988e67f7b039"
        },
        {
          "rel": "versions",
          "type": "application/vnd.grid5000.collection+json",
          "href": "/sites/tacc/versions"
        },
        {
          "rel": "jobs",
```

(continues on next page)
"type": "application/vnd.grid5000.collection+json",
"href": "/sites/tacc/jobs"
},
{
"rel": "deployments",
"type": "application/vnd.grid5000.collection+json",
"href": "/sites/tacc/deployments"
},
{
"rel": "vlans",
"type": "application/vnd.grid5000.collection+json",
"href": "/sites/tacc/vlans"
},
{
"rel": "metrics",
"type": "application/vnd.grid5000.collection+json",
"href": "/sites/tacc/metrics"
},
{
"rel": "status",
"type": "application/vnd.grid5000.item+json",
"href": "/sites/tacc/status"
}
],
{
"description": "University of Chicago",
"email_contact": "help@chameleoncloud.org",
"latitude": 41.718002,
"location": "Argonne National Laboratory, Lemont, Illinois, USA",
"longitude": -87.982952,
"name": "UC",
"security_contact": "help@chameleoncloud.org",
"sys_admin_contact": "help@chameleoncloud.org",
"type": "site",
"uid": "uc",
"user_support_contact": "help@chameleoncloud.org",
"web": "https://www.chameleoncloud.org",
"version": "ee0253a05223dd0f5b88df7f78fb988e67f7b039",
"links": [
{
"rel": "clusters",
"href": "/sites/uc/clusters",
"type": "application/vnd.grid5000.collection+json"
},
{
"rel": "self",
"type": "application/vnd.grid5000.item+json",
"href": "/sites/uc"
},
{
"rel": "parent",
"type": "application/vnd.grid5000.item+json",
"href": "/"
},
{
"rel": "version",
"href": "/sites/uc/versions/ee0253a05223dd0f5b88df7f78fb988e67f7b039"
}]
}
### 12.3.3 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.

For example, to get clusters at TACC:
Again, you will find links in each cluster description. There is a link named nodes for each cluster, which as its name indicates, returns the list of nodes for the specific cluster.

For example, to get nodes on the Alamo cluster at TACC site:

```
curl https://api.chameleoncloud.org/sites/tacc/clusters/alamo/nodes/?pretty
```

You should get back a big collection of nodes. Each node is described in great details, so that you can programmatically find the cluster and nodes that are most suitable for your experiments.

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

```
curl https://api.chameleoncloud.org/sites/tacc/clusters/alamo/nodes/45f0fc6a-a21b-4461-8414-ebf765143aad?pretty | grep -A 10 storage_devices
curl -s https://api.chameleoncloud.org/sites/tacc/clusters/alamo/nodes/0a5b61b2-dc1c-4bee-86f7-247c9689ea88?pretty | grep -A 10 storage_devices
```

### 12.3.4 Fetch the Latest Changes

Let’s go back to the site’s description. In Chameleon, resources are added, updated, or removed over time. If you want to keep an eye on those changes, you can fetch the latest changes that occurred on a specific site:

```
curl https://api.chameleoncloud.org/sites/tacc/versions/?pretty
```

Each version listed in the response represents a change to some resources of the Chameleon testbed.
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.

### 13.1 Provisioning and Managing Resources Using the GUI

To make reservations of the resources, first log into the Horizon web interface - 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, please see Graphical User Interface (GUI).

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

#### 13.1.1 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:** 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.
Fig. 1: The Leases page in the GUI

Fig. 2: The Host Calendar
13.1.2 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.

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. 4: The Create Lease dialog
For specific node reservations, you can find the node UUID using Resource Discovery on the user portal.

Fig. 5: The Create Lease dialog Host reservation tab
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”.

78 Chapter 13. Reservations
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.

*Attention:* 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.

### 13.1.3 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.

**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.

### 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.

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.
Lease Detail

Lease

Name: my-custom-lease
Id: 8ab0cc1f-e2b5-4c51-b9dd-21d2d7007396
Project Id: b6aa718d3e174d06943c89ccaf7bbab3
Start date: 2018-09-03 09:00 BST
End date: 2018-09-03 19:00 BST
Status: PENDING

Events

- **start_lease**
  - Status: Undone
  - Time: 2018-09-03 09:00 BST
- **before_end_lease**
  - Status: Undone
  - Time: 2018-09-03 09:00 BST
- **end_lease**
  - Status: Undone
  - Time: 2018-09-03 19:00 BST

Reservations

status: pending
before_end lease id: default
lease id: 8ab0cc1f-e2b5-4c51-b9dd-21d2d7007396
resource id: 09782cba-fccd-4493-a9ec-6fdd383d7711
max: 1
created_at: 2018-08-30 18:04 BST
min: 1
updated_at: 2018-08-30 18:04 BST
missing_resources: False
hypervisor_properties: [
  
  resource_properties:
    id: d6247519-32b2-4e29-b458-4d079810ccbb

  resource_type: physical:host
resources Changed: False
Fig. 8: The Update Lease Parameters dialog
13.1.4 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, please see Resource discovery. In the Create Lease dialog, select uid in the Resource Type dropdown. Then, choose the UUID of the node you would like to reserve.

13.2 Provisioning and Managing Resources Using the CLI

The sections above present the most user friendly mode of usage, with most actions performed via the web interface. 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.

13.2.1 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:

```
```  

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.

### 13.2.2 Creating a Lease to Reserve Physical Hosts

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

- `--physical-reservation` with the `min`, `max`, 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

For example, the following command will create a lease with the name of `my-first-lease` and the node type of `compute_haswell` that starts on June 17th, 2015 at 4:00pm and ends on June 17th, 2015 at 6:00pm:

```
blazar lease-create \
  --physical-reservation min=1,max=1,resource_properties='["=", "$node_type", \n  "compute_haswell"]' \
  --start-date "2015-06-17 16:00" \
  --end-date "2015-06-17 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:
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:

```
blazar lease-create
   --physical-reservation min=1,max=1,resource_properties='["and", ["="", "$uid", "c9f98cc9-25e9-424e-8a89-002989054ec2"], ["="", "$architecture.smt_size", "48"]]',
   --start-date "2015-06-17 16:00"
   --end-date "2015-06-17 18:00"
   my-custom-lease
```

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

```
blazar lease-create
   --physical-reservation min=1,max=1,resource_properties='["and", ["="", "$uid", "c9f98cc9-25e9-424e-8a89-002989054ec2"], ["="", "$node_type", "compute_haswell"]]',before_end=email
   --start-date "2015-06-17 16:00"
   --end-date "2015-06-17 18:00"
   my-custom-lease
```

```
Currently supported before_end action types include

<table>
<thead>
<tr>
<th>Action Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>email</td>
<td>Send an email notification.</td>
</tr>
<tr>
<td>default</td>
<td>Default action used when no action is specified; Currently set to email.</td>
</tr>
<tr>
<td>''</td>
<td>Do nothing.</td>
</tr>
</tbody>
</table>
```

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:

```
blazar host-list
```

The output should look like:

```
+-----------------+-----------------+---------+---------+-----------------------+
<table>
<thead>
<tr>
<th>id</th>
<th>hypervisor_hostname</th>
<th>vcpus</th>
<th>memory_mb</th>
<th>local_gb</th>
</tr>
</thead>
<tbody>
<tr>
<td>151</td>
<td>00401ba8-4fb0-4f1e-a7dc-e93065ebdd15</td>
<td>24</td>
<td>128000</td>
<td>200</td>
</tr>
<tr>
<td>233</td>
<td>004c89fa-ff13-4563-9012-f2d62c1a7aff</td>
<td>24</td>
<td>128000</td>
<td>200</td>
</tr>
<tr>
<td>330</td>
<td>01029fb8-0a0b-4949-92b0-a756fb8588e5</td>
<td>24</td>
<td>128000</td>
<td>200</td>
</tr>
<tr>
<td>146</td>
<td>036b163e-9fa6-442c-8e6d-cfe12ed5c8a3</td>
<td>24</td>
<td>128000</td>
<td>200</td>
</tr>
<tr>
<td>992</td>
<td>05dd5e25-440f-4492-b3b8-9d39af83b8bc</td>
<td>8</td>
<td>3200</td>
<td>100</td>
</tr>
<tr>
<td>219</td>
<td>066d92f5-7cb9-49ea-8f05-842566672ef</td>
<td>24</td>
<td>128000</td>
<td>200</td>
</tr>
<tr>
<td>3216</td>
<td>06b164d5-3514-4ebe-8928-0bd2f9508b80</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```

(continues on next page)
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:

```
blazar host-show 151
```

The output should look like:

```
+----------------------------------+---------------------------------------------+
| Field                                | Value                                      |
+----------------------------------+---------------------------------------------+
| architecture.platform_type          | x86_64                                    |
| architecture.smp_size              | 2                                          |
| architecture.smt_size              | 48                                         |
| bios.release_date                  | 03/09/2015                                |
| 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                         | 2015-06-26 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.

### 13.2.3 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:

```
blazar lease-update --prolong-for "id" my-first-lease
```
13.2.4 Chameleon Node Types

The following node types are reservable on Chameleon.

| Node Type            | resource_properties='"=', "$node_type", "<Chameleon node type name>"
|----------------------|--------------------------------------------------
| Haswell compute nodes| compute_haswell                                    
| 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                                             

13.2.5 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, 2015 at 4:00pm and ends on June 17th, 2015 at 6:00pm:

```
blazar lease-create --reservation resource_type=network, network_name="my-network" --start-date "2015-06-17 16:00" --end-date "2015-06-17 18:00" my-first-vlan-lease
```

Adding the `network_description` attribute provides its value as the description field when creating the Neutron network, allowing to leverage Chameleon **Software Defined Networking** features.

```
blazar lease-create --reservation resource_type=network, network_name="my-network", network_description="OFController=${OF_CONTROLLER_IP}:${OF_CONTROLLER_PORT}" --start-date "2015-06-17 16:00" --end-date "2015-06-17 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`.
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 Haswell compute node and one VLAN segment:

```
blazar lease-create --physical-reservation min=1,max=1,resource_properties='["==","$node_type","compute_haswell"]' --reservation resource_type=network,network_name="my-network" --start-date "2015-06-17 16:00" --end-date "2015-06-17 18:00" my-combined-lease
```

### 13.2.6 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 omitted, 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, 2015 at 4:00pm and ends on June 17th, 2015 at 6:00pm and reserves three floating IPs:

```
pip install python-openstackclient
PUBLIC_NETWORK_ID=$(openstack network show public -c id -f value)
blazar lease-create --reservation resource_type=virtual:floatingip,network_id=$PUBLIC_NETWORK_ID,amount=3 --start-date "2015-06-17 16:00" --end-date "2015-06-17 18:00" my-first-fip-lease
```
Before launching an instance, make sure you own a lease. About how to create a lease, please 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.

### 14.1 Launching instances with the GUI

#### 14.1.1 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 Flavor step, select the baremetal flavor by clicking the “up” arrow next to it. This is the only flavor available.

**Hint:** If you are familiar with Openstack, other implementations allow for the selection of flavors based on machine disk size and RAM. On Chameleon, the only flavor available is “baremetal” because hardware selection is performed in reservations.

6. 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, please see Networking.
Fig. 1: The Instances page

Fig. 2: The Launch Instance wizard.
Fig. 3: The Source configuration step

Fig. 4: The Flavor configuration step

14.1. Launching instances with the GUI
7. 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](image)

**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.

8. Optionally, you may configure **Scheduler Hints**. This is useful if you would like to launch an instance on a specific node in your multi-node reservation by **UUID**.

   - In the **Custom** text box, type `query` and click the + button. This will add a `query` hint to the list on the right.

   - In the `query` hint, enter your scheduler hint. For example, if you require a specific node, type `\[="\, "$hypervisor_hostname", "$<node_uuid>"\]` where `<node_uuid>` is the node you are requesting.

9. If you want to customize your instance after it has launched, you can 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**.
14.1. Launching instances with the GUI

Fig. 6: Adding a Scheduler Hint
**Tip:** You can disable and turn off appliance agents using a customization script.
10. 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.

![An Instance with the Build status](image)

Fig. 8: An Instance with the Build status

11. After a few minutes, the instance should become *Active*. The power state will show as *Running*. You can now **Associate a Floating IP**.

![An Instance with the Active status](image)

Fig. 9: An Instance with the Active status

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

### 14.1.2 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
### gnocchi_instance

<table>
<thead>
<tr>
<th>Name</th>
<th>gnocchi_instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>b6ecb2b6-d912-46a8-b5c7-6a060e794097</td>
</tr>
<tr>
<td>Status</td>
<td>Build</td>
</tr>
<tr>
<td>Availability Zone</td>
<td>blazar_0ade6d6a-ac39-4ba7-ab07-d27281a58346</td>
</tr>
<tr>
<td>Created</td>
<td>March 15, 2018, 2:39 p.m.</td>
</tr>
<tr>
<td>Time Since Created</td>
<td>0 minutes</td>
</tr>
</tbody>
</table>

#### Specs

<table>
<thead>
<tr>
<th>Flavor Name</th>
<th>baremetal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flavor ID</td>
<td>baremetal</td>
</tr>
</tbody>
</table>

#### IP Addresses

<table>
<thead>
<tr>
<th>Network</th>
<th>IPv4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sharednet 1</td>
<td>10.140.81.88</td>
</tr>
</tbody>
</table>

#### Security Groups

<table>
<thead>
<tr>
<th>Group</th>
<th>Rules</th>
</tr>
</thead>
</table>
| default | ALLOW IPv6 from default  
ALLOW IPv6 to ::/0  
ALLOW IPv4 to 0.0.0.0/0  
ALLOW IPv4 from default |

#### Metadata

<table>
<thead>
<tr>
<th>Key Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Name</td>
<td>CC-CentOS7</td>
</tr>
<tr>
<td>Image ID</td>
<td>785a2bc7-4f4d-4798-8915-4c064b9f46af</td>
</tr>
</tbody>
</table>

#### Volumes Attached

<table>
<thead>
<tr>
<th>Volume</th>
<th>No volumes attached.</th>
</tr>
</thead>
</table>

---

Fig. 10: Instance details
Floating IP dialog. In this dialog, you may simply click Allocate IP. You can optionally specify a description for the IP for your convenience.

Fig. 11: The Allocate Floating IP dialog

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.

Fig. 12: The Floating IP list view with a Floating IP available

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.

14.2 Launching Instances with the CLI

Tip: Reading Command Line Interface (CLI) is highly recommended before continuing on the following sections.
Fig. 13: The Manage Floating IP Associations dialog with an IP selected

Fig. 14: An instance with an allocated Floating IP
14.2.1 Creating an instance with the CLI

To launch an instance inside a reservation, run:

```bash
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:

```bash
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 web interface or by running:

```bash
blazar lease-show <lease_name>
```

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

### 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 web interface in the `Configuration` tab when launching an instance, or by providing a file to the nova command line using the `--user-data` option.

**Tip:** Chameleon supported images are configured with appliance agents, including `collectd` and `Heat agents`. To turn off appliance agents on boot, in order to remove the potential impact on experimental measurements, pass the following script as `user-data`.

```
#!/bin/bash
systemctl stop collectd.service
systemctl disable collectd.service
systemctl stop os-collect-config.service
systemctl disable os-collect-config.service
```

Turning off `collectd` will stop collecting Gnocchi metrics, but you can turn on and configure the daemon anytime for monitoring your experiment.

14.2.2 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.

### 14.2.3 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 please delete them after your experiment is over using:

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

You may find the ID of your ports using:

```
openstack port list
```

### 14.2.4 Launching instances on specific nodes

If you have a reservation for multiple physical nodes, explicitly identified with their UUIDs, you might want to force an instance to be launched on a specific node rather than letting the scheduler select one. This can be done with the CLI using a scheduler hint:

```
openstack server create \
--image CC-CentOS8 \
--flavor baremetal \
--key-name <key_name> \
--nic net-id=<sharednet1_id> \
--hint reservation=<reservation_id> \
--hint query='["=","$hypervisor_hostname","<node_uuid>"]' \
<instance_name>
```

From within an instance you have already launched, you can discover which node it is running on by executing
This will return a JSON dictionary describing site, cluster, and node.

### 14.2.5 Customizing networking

In its default configuration, the bare metal deployment system used by Chameleon (OpenStack Ironic) 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.

### 14.3 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.

#### 14.3.1 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:

```bash
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:

```bash
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.
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, please wait before retrying. Also make sure that you use the cc account. If after 10 minutes you still cannot connect to the machine, please 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.

### 14.3.2 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. 15: The serial console button](image)

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.
14.3. Interacting with instances

Fig. 16: An open console
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 from the Appliance Catalog.

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

## 15.1 The `cc-snapshot` Utility

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](https://github.com).

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

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

In addition, you can exclude directories by using the `-e` flag.
Chameleon Cloud Documentation, Release 2.0a

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

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

You will be prompted to enter your username and password.

**Tip:** You can skip entering username and password by setting the `OS_USERNAME` and `OS_PASSWORD` environment variables. You can set those environment variables manually or using *The OpenStack RC Script*.

**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, please 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`.

**Error:** If you receive the following error:

```
public endpoint for image service in regionOne not found Unable to contact Glance,
```

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:

```bash
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
```

15.2 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, and is part of the Appliance Catalog.

**Tip:** Select *Details* from the dropdown menu to the right of any Chameleon supported appliance to view the relevant
entry from the Chameleon Appliance Catalog.

**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.

### 15.2.1 Uploading an Image

Use the + 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.

### 15.2.2 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.

**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.
Fig. 2: The Create Image dialog
15.2.3 Viewing Image Details

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

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.

15.2.4 Publishing Images to the Appliance Catalog

The dropdown menu to the right of listed images allows their owners to publish an appliance to the Appliance Catalog. Select Publish to Appliance Catalog.

The Create Appliances web form will open automatically with most fields pre-populated. Complete the form and select Create an Appliance.

Entering a descriptive name, author and support contact information, the version, and an informative description can be helpful and 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?

15.3 Managing Images using the CLI

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

15.3.1 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.

15.3.2 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.

15.3.3 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.
15.3.4 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 its `UUID`.

15.3.5 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.

15.3.6 Editing an Image

You may edit an image using the command:

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

Replace `image` with either an image name or its `UUID`. You must provide additional flags to update an image. Use `openstack image set --help` to see all the options.
Chameleon collects monitoring information, representing qualities such as CPU load or power consumption data, from various sources into an aggregation service. Data is kept in this service with resolution that decreases over time. Users can retrieve those metrics via a command line interface (CLI).

In Chameleon, the aggregation service is implemented using the Gnocchi time series database. All Chameleon supported images, from which most of our user’s images are derived, are configured to send a selection of system metrics using the collectd system statistics collection daemon. There is a wide range of qualities this daemon can gather; by default only selected metrics are sent but users can configure the daemon (see Configuring collectd) to adapt this set anytime to monitor their experiments better. Another source of metrics is the infrastructure itself, for example the energy and power consumption metrics.

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

16.1 Setting up the Gnocchi CLI

In addition to installing the CLI, you must also install the Gnocchi client plugin. To install on your local machine, run the following command:

```
pip install gnocchiclient
```

Then, set up your environment for OpenStack command line usage, as described in The OpenStack RC Script.

16.2 Retrieving Metrics

Now, you can run the gnocchi command line utility. To show the different kinds of metrics collected for a specific instance, run:
Chameleon Cloud Documentation, Release 2.0a

```bash
gnocchi resource show <instance_id>
```

**Tip:** You can get the instance' ID from the GUI.

You can get your list of instances by running:

```bash
gnocchi resource list
```

It will print out a chart similar to below:

<table>
<thead>
<tr>
<th>id</th>
<th>type</th>
<th>project_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>8d643431-9a90-4100-8e00-f43d56a68d1e</td>
<td>generic</td>
<td>None</td>
</tr>
<tr>
<td>39ff85e4-cf4e-4969-9408-af47a372ad06</td>
<td>generic</td>
<td>None</td>
</tr>
<tr>
<td>3c6c81ba-0566-4cde-a8c5-7ae4d644293</td>
<td>generic</td>
<td>None</td>
</tr>
<tr>
<td>219f2fec-0e90-4e04-a5d7-1a78c9fde93b</td>
<td>generic</td>
<td>None</td>
</tr>
<tr>
<td>57f2ba05-e57c-4241-bd27-bf95cca9c027</td>
<td>generic</td>
<td>None</td>
</tr>
<tr>
<td>a0cc7bb7-0169-4973-8d4a-08151c52dec6</td>
<td>generic</td>
<td>None</td>
</tr>
<tr>
<td>a6f1d1e2-85db-463c-9769-2a752eb447e</td>
<td>generic</td>
<td>None</td>
</tr>
<tr>
<td>87e52c8d-6c66-43f5-b9fc-da376eccdf2d</td>
<td>generic</td>
<td>None</td>
</tr>
<tr>
<td>bf383c17-d76a-4e50-b347-426c96020d3b</td>
<td>generic</td>
<td>None</td>
</tr>
<tr>
<td>9f25d5f4-79f5-4c34-86b6-79767b8db086</td>
<td>generic</td>
<td>None</td>
</tr>
<tr>
<td>4b8e1ce-9733-4808-921f-6d8ca92a6752</td>
<td>generic</td>
<td>None</td>
</tr>
<tr>
<td>5887a427-286f-47ad-bd4a-d7b9278bbc0f</td>
<td>generic</td>
<td>None</td>
</tr>
<tr>
<td>f5856741-89d5-462f-a0a2-f2423dd9bf38</td>
<td>generic</td>
<td>None</td>
</tr>
<tr>
<td>f7e54e18-9668-4df0-a511-5b2af4c76945</td>
<td>generic</td>
<td>None</td>
</tr>
<tr>
<td>304dc702-c57a-471c-81df-6e711d793e50</td>
<td>generic</td>
<td>None</td>
</tr>
</tbody>
</table>

You will get a result like the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>created_by_project_id</td>
<td>2c8f25efb722467eb9fc25f38996b7c4</td>
</tr>
<tr>
<td>created_by_user_id</td>
<td>7961a8c338ba4cb8a4ac6dfe0ab333f5</td>
</tr>
<tr>
<td>creator</td>
<td>7961a8c338ba4cb8a4ac6dfe0ab333f5:2c8f25efb722467eb9fc25f38996b7c4</td>
</tr>
<tr>
<td>ended_at</td>
<td>None</td>
</tr>
<tr>
<td>id</td>
<td>304dc702-c57a-471c-81df-6e711d793e50</td>
</tr>
<tr>
<td>metrics</td>
<td>interface-enol@if_dropped: 511abf80-d9e9-4e37-bde6-b34de19a7a87</td>
</tr>
<tr>
<td></td>
<td>interface-enol@if_errors: 7bf316e3-ce63-424c-955c-1654541dafe8</td>
</tr>
<tr>
<td></td>
<td>interface-enol@if_octets: 0b9a204b-38fd-4b4f-a51a-c25b9b739c5c</td>
</tr>
<tr>
<td></td>
<td>interface-enol@if_packets: a62006be-d45a-4b2c-a201-4f1b4770f43c</td>
</tr>
</tbody>
</table>

(continues on next page)
16.2. Retrieving Metrics
To get all the measurements of a particular metric, run:

```
gnocchi measures show <metric_name> --resource-id <instance_id> --refresh
```

For example, to get measurements of used memory over time for instance d17d5191-af60-4407-9ed2-e3d48e86ac6d, run:

```
gnocchi measures show memory@memory.used --resource-id d17d5191-af60-4407-9ed2-...e3d48e86ac6d --refresh
```

**Tip:** You may notice that each metric has been assigned a **UUID** to it. Therefore, instead of providing **metric name**, you can provide **metric uuid**.

This will show the latest measurements of that metric with granularity set to 1.0, as well as aggregate values (by default, the mean) over one minute and one hour. Other aggregation methods can be used with the **--aggregation** option, such as std, count, min, max and sum. Your results may appear like this:

<table>
<thead>
<tr>
<th>timestamp</th>
<th>granularity</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-12-22T18:00:00+01:00</td>
<td>3600.0</td>
<td>1222193280.0</td>
</tr>
<tr>
<td>2017-12-22T18:01:00+01:00</td>
<td>60.0</td>
<td>1222684672.0</td>
</tr>
<tr>
<td>2017-12-22T18:02:00+01:00</td>
<td>60.0</td>
<td>1222394538.67</td>
</tr>
<tr>
<td>2017-12-22T18:03:00+01:00</td>
<td>60.0</td>
<td>1222147413.33</td>
</tr>
<tr>
<td>2017-12-22T18:04:00+01:00</td>
<td>1.0</td>
<td>1222684672.0</td>
</tr>
<tr>
<td>2017-12-22T18:05:00+01:00</td>
<td>1.0</td>
<td>1222684672.0</td>
</tr>
<tr>
<td>2017-12-22T18:06:00+01:00</td>
<td>1.0</td>
<td>1222684672.0</td>
</tr>
<tr>
<td>2017-12-22T18:07:00+01:00</td>
<td>1.0</td>
<td>1222684672.0</td>
</tr>
<tr>
<td>2017-12-22T18:08:00+01:00</td>
<td>1.0</td>
<td>1222684672.0</td>
</tr>
<tr>
<td>2017-12-22T18:09:00+01:00</td>
<td>1.0</td>
<td>1222684672.0</td>
</tr>
<tr>
<td>2017-12-22T18:10:00+01:00</td>
<td>1.0</td>
<td>1222684672.0</td>
</tr>
<tr>
<td>2017-12-22T18:11:00+01:00</td>
<td>1.0</td>
<td>1222684672.0</td>
</tr>
<tr>
<td>2017-12-22T18:12:00+01:00</td>
<td>1.0</td>
<td>1222684672.0</td>
</tr>
<tr>
<td>2017-12-22T18:13:00+01:00</td>
<td>1.0</td>
<td>1222684672.0</td>
</tr>
<tr>
<td>2017-12-22T18:14:00+01:00</td>
<td>1.0</td>
<td>1222684672.0</td>
</tr>
<tr>
<td>2017-12-22T18:15:00+01:00</td>
<td>1.0</td>
<td>1222684672.0</td>
</tr>
<tr>
<td>2017-12-22T18:16:00+01:00</td>
<td>1.0</td>
<td>1222684672.0</td>
</tr>
<tr>
<td>2017-12-22T18:17:00+01:00</td>
<td>1.0</td>
<td>1222684672.0</td>
</tr>
<tr>
<td>2017-12-22T18:18:00+01:00</td>
<td>1.0</td>
<td>1222684672.0</td>
</tr>
<tr>
<td>2017-12-22T18:19:00+01:00</td>
<td>1.0</td>
<td>1222684672.0</td>
</tr>
<tr>
<td>2017-12-22T18:20:00+01:00</td>
<td>1.0</td>
<td>1222684672.0</td>
</tr>
<tr>
<td>2017-12-22T18:21:00+01:00</td>
<td>1.0</td>
<td>1222684672.0</td>
</tr>
<tr>
<td>2017-12-22T18:22:00+01:00</td>
<td>1.0</td>
<td>1222684672.0</td>
</tr>
<tr>
<td>2017-12-22T18:23:00+01:00</td>
<td>1.0</td>
<td>1222684672.0</td>
</tr>
</tbody>
</table>

By default, metrics are stored with an archive policy set to “high”, which is defined to keep data as:

- Per second granularity for the last hour
- Per minute granularity for the last week
- Per hour granularity for a year

However, note that since **collectd** is configured to collect metrics only every 10 seconds, there is no metric measurement for each second but every 10 seconds.
16.2.1 Configuring collectd

While only a few `collectd` plugins are enabled by default, you can leverage the large collection of available plugins. To enable a plugin on your instance, edit the instance’s `/etc/collectd.conf` file. Uncomment each `LoadPlugin <plugin_name>` line that you wish to enable. Then, restart collectd with the command:

```
sudo systemctl restart collectd
```

The `collectd` configured to send measurements by batch to minimize network traffic. However, if you want to avoid any interference during your experiments, you can disable `collectd` with the command:

```
sudo systemctl stop collectd && sudo systemctl disable collectd
```

16.2.2 Metrics for bare metal nodes

Chameleon automatically collects power usage and temperature data on all nodes in the system. Instantaneous power usage data (in watts) and temperature readings (in Celsius) are collected through the IPMI interface on the chassis controller for the nodes. This “out-of-band” approach does not consume additional power on the node itself and runs even when the node is powered off.

**Attention:** Temperature metrics are currently collected from the CPU sensor on each node. These temperature readings are only reported while the node is powered on.

As with the system metrics, retrieving these automatically collected metrics for a node requires the OpenStack CLI and Gnocchi client plugin (see installation instructions Setting up the Gnocchi CLI above). To get a list of metrics available for a node, use this command:

```
$ gnocchi resource show <node_uuid>
```

To retrieve a specific reading:

```
$ gnocchi measures show <reading-name> --resource-id=<node_uuid> --refresh
```

**Tip:** The node UUID and the instance UUID are different. You can get a node’s UUID for a reservation from the Horizon GUI (https://chi.tacc.chameleoncloud.org for TACC reservations, https://chi.uc.chameleoncloud.org for UC reservations). Click on your lease name from within the list of leases on the Leases subtab within the Reservations tab. The node UUID is at the very bottom under the Nodes section. You can also find an individual instance node UUID on the instance details page. Click on your instance name on the Instances tab and see Physical Host Name

For example, issuing the following command:

```
$ gnocchi measures show power --resource-id=05dd5e25-440f-4492-b3b8-9d39af83b8bc --refresh
```

returns the following power results for node with id `05dd5e25-440f-4492-b3b8-9d39af83b8bc`. The output below has been truncated:

```
+---------------------------+-------------+--------------------+
| timestamp                | granularity | value              |
+---------------------------+-------------+--------------------+
| 2018-03-21T07:00:00-05:00 | 3600.0      | 3.6990394736842047 |
```

(continues on next page)
To retrieve a metric for a specific time interval, pass the `start` and `stop` parameters; for example:

```bash
$ gnocchi measures show temperature_cpu --start 2018-11-27T02:00:00 --stop 2018-11-27T03:00:00 --resource-id=f3f47a67-d805-48d4-9584-f0143ae976cf --refresh
```

returns:

```
+---------------------------+-------------+---------------+
| timestamp | granularity | value |
+---------------------------+-------------+---------------+
| 2018-11-27T02:00:00-06:00 | 300.0 | 61.0 |
| 2018-11-27T02:05:00-06:00 | 300.0 | 61.0 |
| 2018-11-27T02:10:00-06:00 | 300.0 | 61.0 |
| 2018-11-27T02:15:00-06:00 | 300.0 | 61.0 |
| 2018-11-27T02:20:00-06:00 | 300.0 | 58.6 |
| 2018-11-27T02:25:00-06:00 | 300.0 | 56.5333333333 |
| 2018-11-27T02:30:00-06:00 | 300.0 | 56.0 |
| 2018-11-27T02:35:00-06:00 | 300.0 | 56.0 |
| 2018-11-27T02:40:00-06:00 | 300.0 | 56.0 |
| 2018-11-27T02:45:00-06:00 | 300.0 | 56.0 |
| 2018-11-27T02:50:00-06:00 | 300.0 | 56.0 |
| 2018-11-27T02:55:00-06:00 | 300.0 | 56.0 |
```

### 16.2.3 Energy and Power Consumption Measurement with `etrace2`

The CC-CentOS7, CC-CentOS8, CC-Ubuntu16.04 and CC-Ubuntu18.04 appliances, as well as all Chameleon supported images derived from them, now include support for reporting energy and power consumption of each CPU socket and of memory DIMMs. It is done with the `etrace2` utility which relies on the Intel RAPL (Running Average Power Limit) interface.

**Attention:** Currently, `etrace2` requires a kernel feature that is not supported on our ARM nodes.

To spawn your program and print energy consumption:
etrace2 <your_program>

To print power consumption every 0.5 second:

etrace2 -i 0.5 <your_program>

To print power consumption every 1 second for 10 seconds:

etrace2 -i 1.0 -t 10

For example, to report energy consumption during the generation of a large RSA private key:

```bash
$ etrace2 openssl genrsa -out private.pem 4096
# ETRACE2_VERSION=0.1
Generating RSA private key, 4096 bit long modulus
......................................................................................
˓
→...............................................................................
˓
→...............................................................................
˓
→...............................................................................
˓
→........................................................................++++

e is 65537 (0x10001)
# ELAPSED=2.579472
# ENERGY=365.788208
# ENERGY_SOCKET0=99.037841
# ENERGY_DRAM0=78.577698
# ENERGY_SOCKET1=109.230103
# ENERGY_DRAM1=80.336548
```

The energy consumption is reported in joules.

etrace2 reports power and energy consumption of CPUs and memory of the node during the entire execution of the program. This will include consumption of other programs running during this period, as well as power and energy consumption of CPUs and memory under idle load.

Note the following caveats:

- Intel documents that the RAPL is not an analog power meter, but rather uses a software power model. This software power model estimates energy usage by using hardware performance counters and I/O models. Based on their measurements, they match actual power measurements.

- In some situations the total ENERGY value is incorrectly reported as a value equal or close to zero. However, the sum of ENERGY.Socket and ENERGY.Dram values should be accurate.

- Monitoring periods larger than 10-15 minutes may be inaccurate due to RAPL registers overflowing if they're not read regularly.

This utility was contributed by Chameleon user Kazutomu Yoshii of Argonne National Laboratory.

**Note:** The Linux kernel version of CC-Ubuntu16.04 is too old to use etrace2 on Chameleon Skylake nodes. To solve the problem, simply upgrade the Linux kernel.
CHAPTER 17

Complex appliances

17.1 Introduction

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”.

121
17.2 Complex Appliances in the Catalog

The Chameleon Appliance Catalog 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. Complex Appliances are identified by a badge in their top-right corner representing a group of machines, as in the screenshot below:

![Complex Appliance with a badge](image)

Fig. 1: A Complex Appliance with a badge in the upper right

To view the details of a Complex Appliance, simply click on it.

**Tip:** You may download the Template file or copy the Template file URL to clipboard by clicking the Get Template button. The Template file or its URL is required when launching a Complex Appliance.

17.3 Managing Complex Appliances using the GUI

Before launching a Complex Appliance, please 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 from the Appliance Catalog. At CHI@TACC site or CHI@UC site, go to Project > Orchestration > Stacks use the navigation side bar.

**Tip:** You can go to Stacks page directly from the Appliance Catalog.

1. Go to the Appliance Catalog and identify the appliance you want to launch. Click on it to open its details page.
17.3. Managing Complex Appliances using the GUI

Fig. 2: A Complex Appliance page

Fig. 3: The Stacks page
2. Click on the “Launch Complex Appliance at CHI@TACC” or “Launch Complex Appliance at CHI@UC” button depending on where your reservation is created.

### 17.3.1 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.

![Select Template](image)

**Fig. 4: 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.

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”.

---

124 Chapter 17. Complex appliances
Fig. 5: The Launch Stack step

Fig. 6: A Complex Appliance with the Create in Progress status
17.3.2 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.

**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.
17.3. Managing Complex Appliances using the GUI

**Fig. 8: The Overview tab**

**Fig. 9: The Resources tab**
Fig. 10: The Events tab

Fig. 11: The Template tab
17.3.3 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.

17.4 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:

```bash
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:

```bash
openstack stack list
```

The output should look like the following:

```
+--------------------------------------+---------------+-------------------+----------
<table>
<thead>
<tr>
<th>ID</th>
<th>Stack Name</th>
<th>Stack Status</th>
<th>Creation</th>
</tr>
</thead>
<tbody>
<tr>
<td>e5df33b5-5282-4935-8097-973328ca71e5</td>
<td>my_stack</td>
<td>CREATE_COMPLETE</td>
<td>2018-01-</td>
</tr>
<tr>
<td>23T22:45:12Z</td>
<td>None</td>
<td></td>
<td>23T22:45:12Z</td>
</tr>
</tbody>
</table>
```

17.4.1 Launching a Complex Appliance

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

```bash
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 lists the following parameters section:

```
parameters:
  nfs_client_count:
    type: number
    description: Number of NFS client instances
    default: 1
    constraint:
      range: [ min: 1 ]
      description: There must be at least one client.
  key_name:
```

(continues on next page)
Therefore, in order to use this Template, you must provide values for `nfs_client_count`, `key_name` and `reservation_id`.

### 17.4.2 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:

<table>
<thead>
<tr>
<th>output_key</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>client_ips</td>
<td>Private IP addresses of the NFS clients</td>
</tr>
<tr>
<td>server_ip</td>
<td>Public IP address of the NFS server</td>
</tr>
</tbody>
</table>

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

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

### 17.4.3 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.

### 17.5 Heat Orchestration Templates

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

#### 17.5.1 A Case Example: NFS Share

Let’s look at the **NFS Share Template**. 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_ip_association**: 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.

### 17.5.2 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: it is the Hello World complex appliance.

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.

**Confirm Delete Stacks**

You have selected “my-nfs-cluster”. Please confirm your selection. This action cannot be undone.

![Fig. 12: Confirm deleting stack dialog](image)

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**
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
    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
```

(continues on next page)
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
  description: ID of the Blazar reservation to use for launching instances.
  constraints:
    - custom_constraint: blazar.reservation

Download this template 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:

```yaml
nfs_clients:
  type: OS::Heat::ResourceGroup
  properties:
    count: 2
  resource_def:
    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] }
```

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.

```yaml
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:

```yaml
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 web interface. Of course IP addresses will be specific to each deployment.

![Outputs](image)

Fig. 14: 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.

### 17.5.3 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, please 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 web interface. 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:

```yaml
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.
17.5.4 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 setup 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
  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'] |
| 92a347a9-31a5-43c1-80e2-9cd3e8ebf66f | 192.5.87.224         | [u'reservation:5f470c97-0166-4934-a813-509b743e2d62', u'blazar'] |
| c8480d67-533d-4f55-a197-8271da6d9344 | 192.5.87.71          | []                     |
```

17.5. Heat Orchestration Templates
17.6 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! The process is very similar to regular appliances: log into the Chameleon portal, go to the appliance catalog, and click on the button in the top-right corner: Add an appliance (you need to be logged in to see it).

![Add an appliance](image)

Fig. 15: The Add an Appliance button

You will be prompted to enter a name, description, and documentation. Instead of providing appliance IDs, copy your template to the dedicated field. Finally, share your contact information and assign a version string to your appliance. Once submitted, your appliance will be reviewed. We will get in touch if a change is needed, but if it’s all good we will publish it right away!

17.7 Advanced Topics

17.7.1 All-to-All Information Exchange

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: 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.

This section presents a more advanced form of contextualization that can perform this kind of information exchange. This is implemented by Heat agents running inside instances and communicating with the Heat service to send and receive information. This means you will need to use an image bundling these agents. Currently, all Chameleon-supported images (CC) are supporting this mode of contextualization. If you build your own images using the CC-CentOS7 builder, CC-CentOS builder or CC-Ubuntu builder, you will automatically have these agents installed. This contextualization is performed with several Heat resources:

- OS::Heat::SoftwareConfig: This resource describes code to run on an instance. It can be configured with inputs and provide outputs.

- OS::Heat::SoftwareDeployment: This resource applies a SoftwareConfig to a specific instance.
• **OS::Heat::SoftwareDeploymentGroup**: This resource applies a SoftwareConfig to a specific group of instances.

The template below illustrates how it works. It launches a group of instances that will automatically populates their `/etc/hosts` file with IP and hostnames from other instances in the deployment.

```
heat_template_version: 2015-10-15

description: >
    This template demonstrates how to exchange hostnames and IP addresses to populate /etc/hosts.

descriptors:

parameters:

    flavor:
        type: string
        default: baremetal
        constraints:
            - custom_constraint: nova.flavor

    image:
        type: string
        default: CC-CentOS8
        constraints:
            - custom_constraint: glance.image

    key_name:
        type: string
        default: default
        constraints:
            - custom_constraint: nova.keypair

    instance_count:
        type: number
        default: 2

    reservation_id:
        type: string
        description: ID of the Blazar reservation to use for launching instances.
        constraints:
            - custom_constraint: blazar.reservation

resources:

    export_hosts:
        type: OS::Heat::SoftwareConfig
        properties:
            outputs:
                - name: hosts
                  group: script
                  config:
                      #!/bin/sh
                      (echo -n $(facter ipaddress); echo -n ' '; echo $(facter hostname)) > ${heat_
                      -outputs_path}.hosts

    export_hosts_sdg:
        type: OS::Heat::SoftwareDeploymentGroup
        properties:
            config: { get_resource: export_hosts }
            servers: { get_attr: [server_group, refs_map] }
            signal_transport: HEAT_SIGNAL

    populate_hosts:
        type: OS::Heat::SoftwareConfig
```

(continues on next page)
properties:
  inputs:
    - name: hosts
  group: script
  config: |
    #!/usr/bin/env python
    import ast
    import os
    import string
    import subprocess
    hosts = os.getenv('hosts')
    if hosts is not None:
      hosts = ast.literal_eval(string.replace(hosts, '\n', '\n'))
    with open('/etc/hosts', 'a') as hosts_file:
      for ip_host in hosts.values():
        hosts_file.write(ip_host.rstrip() + '\n')

populate_hosts_sdg:
  type: OS::Heat::SoftwareDeploymentGroup
  depends_on: export_hosts_sdg
  properties:
    config: { get_resource: populate_hosts }
    servers: { get_attr: [server_group, refs_map] }
    signal_transport: HEAT_SIGNAL
    input_values:
      hosts: { get_attr: [export_hosts_sdg, hosts] }

server_group:
  type: OS::Heat::ResourceGroup
  properties:
    count: { get_param: instance_count }
    resource_def:
      type: OS::Nova::Server
      properties:
        flavor: { get_param: flavor }
        image: { get_param: image }
        key_name: { get_param: key_name }
        networks:
          - network: sharednet1
        scheduler_hints: { reservation: { get_param: reservation_id } }
        user_data_format: SOFTWARE_CONFIG
        software_config_transport: POLL_SERVER_HEAT

outputs:
  deployment_results:
    value: { get_attr: [export_hosts_sdg, hosts] }

There are two SoftwareConfig resources:

- The first SoftwareConfig, export_hosts, uses the facter tool to extract IP address and hostname into a single line (in the format expected for /etc/hosts) and writes it to a special path (${heat_outputs_path}.hosts). This prompts Heat to assign the content of this file to the output with the name hosts.

- The second SoftwareConfig, populate_hosts, takes as input a variable named hosts, and applies a script that reads the variable from the environment, parses it with ast.literal_eval (as it is formatted as a Python dict), and writes each value of the dictionary to /etc/hosts.
The SoftwareDeploymentGroup resources `export_hosts_sdg` and `populate_hosts_sdg` apply each SoftwareConfig to the instance ResourceGroup with the correct configuration.

Finally, the instance ResourceGroup is configured so that each instance uses the following contextualization method instead of a user_data script:

<table>
<thead>
<tr>
<th>user_data_format: SOFTWARE_CONFIG</th>
</tr>
</thead>
<tbody>
<tr>
<td>software_config_transport: POLL_SERVER_HEAT</td>
</tr>
</tbody>
</table>

You can follow the same template pattern to configure your own deployment requiring all-to-all information exchange.

### 17.7.2 Automated Deployment

On Chameleon you can configure a Heat Stack to launch as soon as your lease begins. Whether your experiments require a large cluster or a single node, automated deployment saves you time configuring your environment and even allows you to run your entire experiment automatically when the necessary resources become available.

At present, you will need to use our customized versions of the Heat and Blazar CLI tools to implement this feature.

#### Install Custom CLI

You can install Chameleon’s `python-heatclient` and `python-blazarclient` packages via `pip` by running the following commands:

```bash
pip install git+https://github.com/ChameleonCloud/python-heatclient.git
pip install git+https://github.com/ChameleonCloud/python-blazarclient.git
```

#### Initialize Stack

Next you will need to configure a Heat stack with the `--initialize` flag on the CLI and a dummy `reservation_id` parameter. The dummy id can be anything (even an empty string) so long as the `reservation_id` parameter is specified so that Blazar can overwrite it once your advanced reservation is scheduled and the stack is ready to launch. Once your stack is initialized, the status should read `INIT_COMPLETE`. This indicates that your template was validated and all the data required to launch a stack has been stored. See example command below:

```bash
openstack stack create -t <template_file> --initialize --parameter reservation_id=dummy <stack_name>
```

#### Create Reservation with Stack_ID

Finally, for a stack to launch when your reservation begins, we need to let Blazar know which stack to notify Heat to update. This is done via the command line by specifying `orchestration` as an `on_start` action with a stack_id (e.g. `on_start=orchestration:<stack_id>`) under the `--physical-reservation` flag. Under the hood, Blazar will update your initialized Heat stack with the reservation_id assigned to the lease. See example command below:

```bash
blazar lease-create --start-date "<start_date>" --end-date "<end_date>" \ 
--physical-reservation min=<min>,max=<max>,on_start=orchestration:<stack_id> \ 
<lease_name>
```
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.

**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.

### 18.1 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 web interfaces under the Object Store panel.

**Hint:** KVM@TACC users can access the TACC store by using their CHI@TACC OpenStack RC file.

### 18.2 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*.

### 18.3 Managing Object Store using the GUI

To access the *Object Store* using the GUI at CHI@TACC or CHI@UC, use the navigation sidebar to go to *Project > Object Store > Containers*.

![Fig. 1: The Containers page](image)

#### 18.3.1 Working with Containers

To create a container, click the *+Container* button. This will open 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.

You may click on a *Container* to see the details and work with *Objects* belong 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*.

#### 18.3.2 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.
18.3. Managing Object Store using the GUI

Fig. 2: The Create Container dialog

A container is a storage compartment for your data and provides a way for you to organize your data. You can think of a container as a folder in Windows® or a directory in UNIX®. The primary difference between a container and these other file system concepts is that containers cannot be nested. You can, however, create an unlimited number of containers within your account. Data must be stored in a container so you must have at least one container defined in your account prior to uploading data.

Fig. 3: The Container list
Fig. 4: Container details

Fig. 5: The Delete Container button

Fig. 6: The Upload button
18.3.3 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.

18.4 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
```
Fig. 9: A Container with a Folder

Fig. 10: A Folder within the Container
Then, you must set environment variables for your account and project using *The OpenStack RC Script*.

### 18.4.1 Working with Containers

To create a *Container*, use the following command:

```bash
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:

```bash
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:

```bash
openstack container show <container_name>
```

To view a list of objects within a container, use the command:

```bash
openstack object list <container_name>
```

To download a container with all the objects belong to it, use the following command:

```bash
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!**

```bash
openstack container delete --recursive <container_name>
```

### 18.4.2 Working with Objects

You may upload a file from your local machine to a container using the following command:

```bash
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:

```bash
openstack object delete <container_name> <object_name>
```

If you wish to download an individual object directly from a container, use the command:

```bash
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

```python
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:

```python
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>
```

18.4.3 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.

18.5 Mounting Object Store as a File System

**Tip:** Cloudfuse can upload objects up to 4GB. For larger objects, please use the Swift CLI.

When logged into an instance using Chameleon-supported images, such as CC-CentOS8 and CC-Ubuntu18.04, you will see a directory called `my_mounting_point` which is a pre-mounted directory to your Chameleon Object Store at the same site of your instance. Each Object Store container that you have access to will appear as a subdirectory inside this mount.

The `cc-cloudfuse` tool (Source: ChameleonCloud/cc-cloudfuse) is pre-installed in Chameleon-supported images. It is based on the `cloudfuse` tool (Source: redbo/cloudfuse), which is used to mount your Chameleon Object Store as a directory on your Linux environment.

**Important:** Some older Chameleon-supported images have an outdated version of this tool installed, which is not compatible with authentication using federated identity. If you wish to continue using a historical image, you should update the tool by following the installation instructions.

To mount, use the following command:

```bash
cc-cloudfuse mount <mount_dir>
```

Now you can access your Chameleon Object Store as your local file system.

To unmount, use the following command:
Important: Limitations

The primary usage scenario of the cc-cloudfuse tool is to allow you to interact with Chameleon Object Store using familiar file system operations. Because the cc-cloudfuse 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.
7. The maximum number of listings is 10,000 items.

Please keep these limitations in mind when evaluating cc-cloudfuse.

Note: You may experience persistence issues when using cc-cloudfuse, especially when writing large files or writing many files at the same time. Unmounting and re-mounting usually resolves this.
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. Theses experiments can utilize Chameleon’s advanced networking capabilities including Isolated Network VLANs, External Layer2 Connections (Stitching), and Software Defined Networking.

19.1 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.

19.1.1 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.

19.1.2 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.
19.1.3 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. Please be responsible and release any unused floating IP address, so other Chameleon users and projects can use them!

## 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.

<table>
<thead>
<tr>
<th>Site</th>
<th>Hostname pattern (for IP AA.BB.CC.DD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHI@TACC</td>
<td>chi-dyn-AA-BB-CC-DD.tacc.chameleoncloud.org</td>
</tr>
<tr>
<td></td>
<td>e.g., chi-dyn-129-114-108-147.tacc.chameleoncloud.org</td>
</tr>
<tr>
<td>CHI@UC</td>
<td>chi-dyn-AA-BB-CC-DD.uc.chameleoncloud.org</td>
</tr>
<tr>
<td></td>
<td>e.g., chi-dyn-192-5-87-98.uc.chameleoncloud.org</td>
</tr>
</tbody>
</table>

19.1.4 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 allow all communication within Chameleon private networks, but 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 `ufw` enabled and the following rules set:

```
ufw limit ssh
ufw allow http
ufw allow https
ufw allow from 10.0.0.0/8
ufw allow from 172.16.0.0/12
ufw allow from 192.168.0.0/16
ufw default deny incoming
```

These rules allow ssh traffic on port 22 over the public internet, but will limit the number of attempts to 6 for every 30 seconds.

HTTP/HTTPS on port 80 and 443 are allowed.

Communication is fully allowed over the 3 private network ranges use by Chameleon.

Any other incoming connections will be denied.

You can enable communication over a specific port using the command:

```
sudo ufw allow <port>
```

You can also permit connections from a specific ip or network:

```
sudo ufw allow from <ip address or network>
```

The man page for `ufw` has more examples.

Security Groups

KVM@TACC supports Security Groups, which can be assigned directly to instances upon launch or after the instance is already running. By default, instances have no Security Groups applied, so all traffic is allowed.

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.

19.2 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.

19.2.1 Configuring Networking using a Heat template

1. Go to Project > Orchestration > Stacks.
2. Click the Launch Stack button to open an interactive dialog.
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 address 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.

19.2.2 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.

**Attention:** Do not select the Disable Gateway checkbox!

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.
Fig. 1: The Create Network dialog

Fig. 2: The Subnet tab
Fig. 3: Subnet details
• **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.

![Create Router dialog](image)

**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.

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.
Fig. 5: The Router interface in the Network Topology page

Fig. 6: The Interfaces tab in the Router detail page
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.

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.

19.2.3 Configuring Networking using the CLI
Fig. 8: Dropdown for deleting router

**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*.

**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_zoneHints    |                                      |
| availability_zones        |                                      |
| created_at                | 2018-03-23T23:45:19Z                  |
| description               | None                                 |
| 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                | d5233415ee0b467baecl4cbd2d0e1331      |
| provider:network_type     | vlan                                 |
| provider:physical_network | physnet1                             |
| provider:segmentation_id  | 2018                                 |
+---------------------------+--------------------------------------+

(continues on next page)
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:

```
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-ba49-4c0c-8480-ba43dd4724c2  |
| ip_version        | 4                                    |
| ipv6_address_mode | None                                 |
| ipv6_ra_mode      | None                                 |
| name              | MySubnet                             |
| network_id        | 21ed933c-323d-4708-930c-5df82c507430  |
| project_id        |                                      |
| 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:

```bash
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              | d5233415ee0b467baec14cbb2d0e1331     |
| 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:

```bash
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:

```bash
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:

```bash
openstack router unset --external-gateway <router_name>
openstack router remove subnet <router_name> <subnet_name>
openstack router delete <subnet>
openstack network delete <network_name>
```
19.3 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 extend across Chameleon, potentially other testbeds such as GENI, and into physical resources on their own campus networks. Users can create a dedicated network associated with a dynamic VLAN, subnet with own DHCP server, and router for external connections.

Currently, it is possible to connect user-configured networks to other domains (e.g. GENI) over circuits created on Internet2’s Advanced Layer 2 Service (AL2S). In this setup, a pool of VLANs is extended from Chameleon CHI@UC racks to the AL2S endpoint at StarLight. Currently, 10 VLAN tags (3290-3299) are dedicated to this AL2S endpoint, although 3290 is reserved for system use. A user-configured network that is associated with one of the dedicated AL2S VLAN tags (segmentation ID must be the same as AL2S VLAN tag) can be stitched to external domains (e.g. GENI). A circuit on AL2S needs to be created.

This document describes how to stitch Chameleon experiments to external resources including ExoGENI and Internet2 connected campuses. You will need to know how to create stitchable dynamic VLANs as described in the Isolated Network VLANs documentation. After you have created such VLAN this document will describe how to create a slice in three cases: connect to ExoGENI, connect to other domains using ExoGENI as an intermediary, or connect to other domains directly.

This document also describes connecting isolated stitchable networks across Chameleon sites (CHI@UC and CHI@TACC) over layer-2 circuits.

Chameleon has the capability to create dynamically managed VLANs associated with user-configured private IP subnets as described on Isolated Network VLANs. Users can create a dedicated network associated with a dynamic VLAN, subnet with own DHCP server, and router for external connections. These networks can be created through the web as well as command line interface. User-configured networks (isolated networks) are associated with VLANs by Segmentation IDs.

In the following sections, this workflow is described for different settings.

19.3.1 Configuring a Stitchable Network

In this documentation, we will describe how to stitch to the ExoGENI testbed. 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 using the CLI (See Creating a Lease to Reserve a VLAN Segment). Once you reserve any VLAN segment, your network will be created automatically. To reserve a segment on the appropriate external testbed make sure to include exogeni as the physical_network in the resource_properties attribute. An example is provided below:

```
blazar lease-create --reservation resource_type=network,resource_name=my-stitchable-network,resource_properties='{"="","$physical_network","exogeni"} PRIMARY' --start-date "2015-06-17 16:00" --end-date "2015-06-17 18:00" my-stitchable-network-lease
```

19.3.2 Connecting Chameleon to ExoGENI

ExoGENI is one of the two primary GENI testbeds. ExoGENI allows users to create isolated experimental environments with compute and network resources distributed across 20 sites. ExoGENI has a special type of connection called “stitchport” which is a formally defined meeting point between VLANs dynamically provisioned within Chameleon and ExoGENI slices. Users can create slices on ExoGENI testbed, and connect these slices with Chameleon nodes by using a stitchport.

Stitchports that exist in ExoGENI topology are listed on ExoGENI Wiki (ExoGENI Resource Types: Stitchport Identifiers). URLs for port locations and corresponding VLAN tags are used to create a stitchport connection. Stitchport information for Chameleon is listed as below:
Layer 2 connections on ExoGENI are provisioned on AL2S by an agent that submits requests to OESS on behalf of the user slice. Users do not need to have an OESS account. An ExoGENI slice with a stitchport can be created as below. (Information for using ExoGENI and creating slices can be found at http://www.exogeni.net)

1. Connect the node to a stitchport:

2. Supply the URL and VLAN tag to the stitchport properties. VLAN 3299 will be used for Chameleon connection. The user-configured network on Chameleon must have segmentation ID: 3299.

3. Submit request. A manifest for the reservations will be returned.

4. After the slice creation is completed, nodes in the slice will be able to connect to the Chameleon nodes that are connected to the user-configured network with segmentation ID: 3299.

5. It is also possible to connect ExoGENI nodes to both CHI@UC and CHI@TACC with multiple interfaces.

19.3.3 Connecting Chameleon to user owned domains via ExoGENI

Using ExoGENI to connect to Chameleon can be further extended by using ExoGENI as an intermediary domain.
19.3. External Layer2 Connections (Stitching)
19.3. External Layer2 Connections (Stitching)
In this use case, a local site can be connected to ExoGENI via stitchports, and an ExoGENI slice can be created to route traffic to Chameleon. In the example below, a stitchport connects the local site (NCBI) to an ExoGENI slice which is connected to Chameleon. Nodes on the ExoGENI slice can be used to route traffic from NCBI nodes to Chameleon nodes. In this case, all layer 2 circuits will be provisioned by ExoGENI.

Connecting a local site to ExoGENI via stitchports is a process that requires multiple steps involving site owners, regional network providers, and ExoGENI.

ExoGENI racks are located on campuses across the US. Campuses are connected to Internet2 AL2S via regional provider networks. A set of VLAN tags is reserved for ExoGENI from the pool of available VLAN tags by the regional providers and campus administrations. These VLANs are plumbed on both regional provider and campus networks all the way from AL2S endpoint to the rack or server(s). Some campuses/institutions are directly connected to AL2S nodes without a regional provider (e.g. Pittsburgh Supercomputing Center, George Washington University (CAREEN)).

Stitchports can be used to connect a specific location to ExoGENI racks.

1. VLAN(s) from the local site should be extended through the campus network all the way to the AL2S endpoint.
2. ExoGENI must update the topology to activate the stitchport.

ExoGENI administrators can provide assistance and can be contacted at geni-orca-users@googlegroups.com

19.3.4 Connecting Chameleon to user owned domains

Users can connect their local domains to Chameleon over manually created layer-2 circuits on AL2S. Local domains need to be connected to the other AL2S endpoint of the circuit by users.
Circuits on AL2S are created through the Internet2 AL2S OESS portal. The OESS (Open Exchange Software Suite) is a set of software used to configure and control dynamic layer 2 virtual circuit (VLAN) networks on OpenFlow enabled switches. It includes a web-based user interface as well as a web services API.

Chameleon is connected to the AL2S endpoint at StarLight:

| Node: sdn-sw.star.net.internet2.edu |
| Interface: et-8/0/0 |
| VLAN range: 3290-3299 |

A user can log into the AL2S OESS portal and create a circuit connecting the Chameleon endpoint to the user-owned endpoint. The user should have an account to log in to the AL2S OESS portal. On OESS, users are members of workgroups. After logging in to the portal, a user can see the workgroups that he/she is a member of.

Network resources on AL2S are granted access to the workgroups. This access is granted by the owner of the AL2S network resource (campus network administrators or network engineers at regional providers). After granting access to the resources, they become available for the workgroup and start showing up in the “Available Resources” section. For the user to create such a circuit on AL2S with Chameleon endpoint, the workgroup that the user has membership should be granted access for this endpoint. This can be requested from Chameleon by opening a ticket with our help desk.

As an example, Chameleon resources can be seen in “Available Resources” section for a user in the “ExoGENI” workgroup after access to the workgroup is granted.

The user in the ExoGENI workgroup can create a circuit with two endpoints to connect a local site to Chameleon.

| Endpoint 1 (Local site): |
| Node: sdn-sw.rale.net.internet2.edu |
| Interface: et-9/0/0 |
| VLAN: 3998 |

| Endpoint 2 (UC@UC): |
| Node: sdn-sw.star.net.internet2.edu |
| Interface: et-8/0/0 |
| VLAN: 3290 |

To create a circuit, follow these instructions:

1. Create a new VLAN
2. Select endpoints
3. Submit circuit request
4. When the circuit is provisioned, you should see this:
5. In addition, the Path can be seen on the map. Utilization data becomes available after 3 hours.
Chapter 19. Networking

OS3E The Open Science, Scholarship & Services Exchange

Available Workgroups

Search for Workgroups:

Workgroup:
- EcoGENI
- RENCI

Current Features
- Self-service virtual circuit provisioning web portal
- Integrated per port VLAN usage monitoring
- Sub-second provisioning
- Automatic failover and restoration to and from backup path
- Work groups for shared management of resources
- Admin web interface for service management
- Switch and topology discovery
- IETF-based inter-domain circuit provisioning
- Peering data export

Known Issues

There are several features we are unable to support with OpenFlow 1.0, we anticipate that these will be supported when we transition to OpenFlow 1.3.
- NAC Address Learning
- Q-in-Q
- QinQ (QoS)

Editing of multi-point circuits with a large number of endpoints may fail. Please contact the Internet2 NOC if you need to modify a large multi-point circuit and encounter an issue.
19.3. External Layer2 Connections (Stitching)
19.3. External Layer2 Connections (Stitching)
19.3. External Layer2 Connections (Stitching)
At this point, a layer-2 circuit is created on AL2S. The user-configured network with segmentation ID 3290 can be connected to the local servers. The user needs to extend the VLANs at the local site (3998 in this case) to the AL2S endpoint.

To obtain an account to access AL2S OESS portal, users should contact Internet2. Information can be found from the links below:

- AL2S Participants
- AL2S Layer 2 Service Workgroups
- AL2S FAQ
- Using OESS

### 19.3.5 Connecting Stitchable Isolated Networks across Chameleon Sites

1. Create isolated networks by specifying the “exogeni” provider. Follow the documentation for Configuring a Stitchable Network. A “stitchable” VLAN tag will be returned and “Physical Network” will appear as “Exogeni” on the dashboard. This step will be executed the same way on both UC and TACC sites.

2. After having stitchable isolated networks on UC and TACC sites, a request should be sent to the Help Desk for creation of AL2S circuits. In the request, following information should be specified: - Information for the network at UC (Project ID, name of the network, ID of the network) - Information for the network at TACC (Project ID, name of the network, ID of the network) - Duration of the circuit in active state

### 19.4 Software Defined Networking

**Tip:** A good way to start working with OpenFlow on Chameleon is the OpenFlow Quick Start appliance.

Chameleon’s Bring Your Own Controller (BYOC) functionality enables tenants to create isolated network switches managed using an OpenFlow controller provided by the tenant. This feature is targeted at users wishing to experiment with software-defined networking (SDN) as well as enabling custom network appliances supporting experiments that have non-standard networking requirements. This document focuses on how to use OpenFlow networks on Chameleon. A complete discussion of OpenFlow and SDN is beyond the scope of this document.

**Note:** More information on OpenFlow: https://www.sdxcentral.com/sdn/definitions/what-is-openflow/ and https://www.opennetworking.org/technical-communities/areas/specification/open-datapath/

OpenFlow switches, like traditional switches, forward network traffic between a number of ports used to connect other networks and devices. The primary difference is that OpenFlow switches rely on external software (a “controller”) to dynamically manage the rules (or “flows”) that determine how and where the traffic is forwarded. In addition, OpenFlow enables a much larger set of possible rules which can be imposed on the traffic.

The basic requirements of an OpenFlow switch are the switch and the controller. The switch is configured with the IP address and port of a controller (software) that manages the switch’s rules. When a packet arrives at the switch, the packet is tested against the rules that are known by the switch to determine what action(s) to take. Typically, if there are no rules that apply to a packet, the packet is sent to the controller which replies with a set of rules for that type of packet. The new rules are cached in the switch and applied to subsequent packets until the rules expire or are explicitly removed.
**19.4.1 Chameleon and OpenFlow**

BYOC is part of the expanded deployment for Chameleon’s phase 2. It enables tenants to allocate OpenFlow switches controlled by their own OpenFlow controller. This capability is limited to the phase 2 hardware additions that include the Corsa DP2000 series OpenFlow switches and Skylake compute nodes. The Corsa switches are key to enabling the BYOC functionality. These switches allow for the creation of mutually isolated forwarding contexts which can be thought of as virtual OpenFlow switches even though they are the native abstraction used by the Corsa DP2000s. Each isolated forwarding context can be configured to use its own OpenFlow controller. The Chameleon BYOC functionality enables tenants to specify the IP and port of an arbitrary OpenFlow controller when they create private networks.

**Important:** OpenFlow capabilities are site-dependent. At UC, both Haswell and Skylake node types can participate in an OpenFlow network; at TACC, only Skylakes are supported.

Specifying an OpenFlow controller for your private network is just a special case of creating a private network. Before proceeding you should become familiar with using regular private VLANs on Chameleon and be able to create your own private VLANs. Background information can be found in the document covering Reconfigurable Networking.

**Attention:** Currently it is not possible to specify an OpenFlow controller using the Horizon portal. However, OpenFlow networks with tenant owned controllers can be created using Heat templates which integrate the instructions below.

Using the CLI, an external OpenFlow controller (IP and port) can be specified on the command line using the `--description` field as shown below. Creating the subnet and router is the same as any other Chameleon network.

```
openstack network create --provider-network-type vlan --provider-physical-network exogeni --description OFController=<OF_Controller_IP>:<OF_Controller_Port> <network_name>
```

**Note:** To reserve a stitchable VLAN segment and use it for OpenFlow, you must use the Blazar CLI. See Creating a Lease to Reserve a VLAN Segment.

The output should look like the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin_state_up</td>
<td>UP</td>
</tr>
<tr>
<td>availability_zone_hints</td>
<td></td>
</tr>
<tr>
<td>availability_zones</td>
<td></td>
</tr>
<tr>
<td>created_at</td>
<td>2018-05-23T14:38:18Z</td>
</tr>
<tr>
<td>description</td>
<td>OFController=162.250.136.46:6653</td>
</tr>
<tr>
<td>dns_domain</td>
<td>None</td>
</tr>
<tr>
<td>id</td>
<td>5e359c6f-a69e-4f4d-b92a-784a5f6ca59f</td>
</tr>
<tr>
<td>ipv4_address_scope</td>
<td>None</td>
</tr>
</tbody>
</table>

(continues on next page)
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipv6_address_scope</td>
<td>None</td>
</tr>
<tr>
<td>is_default</td>
<td>None</td>
</tr>
<tr>
<td>mtu</td>
<td>1500</td>
</tr>
<tr>
<td>name</td>
<td>exogeni-exogeni-3294</td>
</tr>
<tr>
<td>port_security_enabled</td>
<td>False</td>
</tr>
<tr>
<td>project_id</td>
<td>e8ae724d28374d0fa15a0e16674b5c47</td>
</tr>
<tr>
<td>provider:network_type</td>
<td>vlan</td>
</tr>
<tr>
<td>provider:physical_network</td>
<td>exogeni</td>
</tr>
<tr>
<td>provider:segmentation_id</td>
<td>3294</td>
</tr>
<tr>
<td>qos_policy_id</td>
<td>None</td>
</tr>
<tr>
<td>revision_number</td>
<td>2</td>
</tr>
<tr>
<td>router:external</td>
<td>Internal</td>
</tr>
<tr>
<td>segments</td>
<td>None</td>
</tr>
<tr>
<td>shared</td>
<td>False</td>
</tr>
<tr>
<td>status</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>subnets</td>
<td></td>
</tr>
<tr>
<td>updated_at</td>
<td>2018-05-23T14:38:18Z</td>
</tr>
</tbody>
</table>

Example CLI command used to create the network:

```
NET="exogeni-3294"
PHYSICAL_NETWORK_TENANT="exogeni"
NET_TYPE="vlan"
NET_NAME="/$(PHYSICAL_NETWORK_TENANT)-$(NET)"
OF_CONTROLLER_IP="162.250.136.46"
OF_CONTROLLER_PORT="6653"
openstack network create --provider-network-type $NET_TYPE \ 
    --provider-physical-network $(PHYSICAL_NETWORK_TENANT) \ 
    --description OFController=$(OF_CONTROLLER_IP):$OF_
    $NETWORK_NAME$ \ 
    $NET_NAME$
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin_state_up</td>
<td>UP</td>
</tr>
<tr>
<td>availability_zone_hints</td>
<td></td>
</tr>
<tr>
<td>availability_zones</td>
<td></td>
</tr>
<tr>
<td>created_at</td>
<td>2018-05-23T14:38:18Z</td>
</tr>
<tr>
<td>description</td>
<td>OFController=162.250.136.46:6653</td>
</tr>
<tr>
<td>dns_domain</td>
<td>None</td>
</tr>
<tr>
<td>id</td>
<td>5e359c6f-a69e-4f4d-b92a-784a5f6ca59f</td>
</tr>
<tr>
<td>ipv4_address_scope</td>
<td>None</td>
</tr>
<tr>
<td>ipv6_address_scope</td>
<td>None</td>
</tr>
<tr>
<td>is_default</td>
<td>None</td>
</tr>
<tr>
<td>mtu</td>
<td>1500</td>
</tr>
<tr>
<td>name</td>
<td>exogeni-exogeni-3294</td>
</tr>
<tr>
<td>port_security_enabled</td>
<td>False</td>
</tr>
<tr>
<td>project_id</td>
<td>e8ae724d28374d0fa15a0e16674b5c47</td>
</tr>
<tr>
<td>provider:network_type</td>
<td>vlan</td>
</tr>
<tr>
<td>provider:physical_network</td>
<td>exogeni</td>
</tr>
<tr>
<td>provider:segmentation_id</td>
<td>3294</td>
</tr>
<tr>
<td>qos_policy_id</td>
<td>None</td>
</tr>
<tr>
<td>revision_number</td>
<td>2</td>
</tr>
<tr>
<td>router:external</td>
<td>Internal</td>
</tr>
</tbody>
</table>
At this point your OpenFlow network switch will have been created and connected to the OpenFlow at the IP/Port that you specified. Using your controller you can explore the OpenFlow switch. There should be only one port on the switch with is the uplink that connects to the OpenStack services and, optionally, any externally stitched networks such as ExoGENI. The uplink port ID will be the segmentation ID (VLAN ID) of the network shown in the Chameleon portal. When nodes are created and connected to your network ports will be added to your OpenFlow switch. Each compute node will always have the same port ID on the switch. The mapping of port IDs to compute nodes is in the following section.

19.4.2 Port Mapping

You will likely need your OpenFlow controller to know which of its ports connects to which of your Chameleon nodes. The uplink port will always match the segmentation ID (VLAN ID) of the network. The following tables list the OpenFlow port numberings for each baremetal node; these port values will be stable for every deployment of a particular node.

CHI@UC

Note: Both compute_haswell and compute_skylake nodes can participate in SDN experiments at CHI@UC. This is not (yet) the case at CHI@TACC; at that site, only compute_skylake nodes can be part of OpenFlow-enabled networks.
<table>
<thead>
<tr>
<th>Node UUID (compute_haswell rack 1)</th>
<th>Port</th>
<th>Node UUID (compute_haswell rack 2)</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>6ad2bf8e-4098-40c2-b68c-05b9d3f72d4f</td>
<td>10221</td>
<td>04d57e8-5239-41ae-b087-7e67f4ab26</td>
<td>10263</td>
</tr>
<tr>
<td>4b8e8bf4-ef7d-40ae-addb-af572f7d3cd4</td>
<td>10222</td>
<td>eb160fa5-eech5-43dd-b76e-3883a4e99184</td>
<td>10267</td>
</tr>
<tr>
<td>ec24b2ef-29b9-45f-88f-3a9de595f0e3</td>
<td>10223</td>
<td>8f4b8129-3c7c-4b47-a8e6-94d578ad3763</td>
<td>10267</td>
</tr>
<tr>
<td>9c5e8be-16f-4e4-4c1-978b-512963be2c9</td>
<td>10224</td>
<td>a6b31d60-0656-46d2-9410-3504f0daae23</td>
<td>10266</td>
</tr>
<tr>
<td>d5a2a225-6312-44e0-8130-87a171a1006d</td>
<td>10225</td>
<td>ddd3615-5212-6123-3ed5d457ea67</td>
<td>10267</td>
</tr>
<tr>
<td>5c6af0a0-11a-48c1ab8-36469e4ab8d7</td>
<td>10226</td>
<td>6bb7c530-cc11-47b5-9e64-136d5d457ea6</td>
<td>10266</td>
</tr>
<tr>
<td>a3365780-9a9b-4c87-8b08-7a107e5abe5b</td>
<td>10227</td>
<td>15b298a-52aa-44d4-8318-7b1a0708eae5</td>
<td>10269</td>
</tr>
<tr>
<td>2f57c329-9461-4ada-b95f-1963e80732f</td>
<td>10228</td>
<td>12985e8f-e3ae-4408-b971-cceef4e75975</td>
<td>10271</td>
</tr>
<tr>
<td>fed77a2-051-412f-95c-45b3e2fbf14d</td>
<td>10229</td>
<td>b11fe2f-2d75-43a-94f-8e092ba8e8</td>
<td>10272</td>
</tr>
<tr>
<td>c03b5a7-6b47-43d-abb9-75ad93482d4</td>
<td>10230</td>
<td>45593d29-91b2-418b-b212-3a333a4d8</td>
<td>10273</td>
</tr>
<tr>
<td>f50dec35-491-4667-a9fa-d30695c5661</td>
<td>10231</td>
<td>cc16f4b-59d-40e-b1b3-1a3b40f87b3</td>
<td>10274</td>
</tr>
<tr>
<td>7ccc312f-3251-428-987-27cada4b957a</td>
<td>10232</td>
<td>572fa12c-18d2-640b-2b2a-e059242aae5</td>
<td>10275</td>
</tr>
<tr>
<td>c2d495ef-76bc-4e4-4c1-978b-512963be2c9</td>
<td>10233</td>
<td>d4bb358e-ad98-4873-90a7-f84da62b804</td>
<td>10276</td>
</tr>
<tr>
<td>806598b-73d4-4c0-879-y44a998bbcb6</td>
<td>10234</td>
<td>5c13b830-58d0-4d4a-4b3a-31aa3a3b53</td>
<td>10277</td>
</tr>
<tr>
<td>9c81036f-b389-4393-982-y6e17638984</td>
<td>10235</td>
<td>05a27a3-3edf-4fa0-a04d-f5e6472e44</td>
<td>10278</td>
</tr>
<tr>
<td>13d98b8e-38c-42c-7b6-8bb474b4ca4c</td>
<td>10236</td>
<td>37d9111d-439b-4a89-a9fa-199378259d1</td>
<td>10279</td>
</tr>
<tr>
<td>60f65717-9b65-4b0-4d5-0ac6e77e587</td>
<td>10237</td>
<td>8445f7e-9ad9-487-90a3-f8dd6bba804</td>
<td>10280</td>
</tr>
<tr>
<td>b05d1c1-11de-4641-9b26-0b21c708b81</td>
<td>10238</td>
<td>9f50c93-d9a-f4da-8e2-8e87f1e2700</td>
<td>10281</td>
</tr>
<tr>
<td>5a4a21f-5c78-4f-49f-43c3f21cd0</td>
<td>10239</td>
<td>b64905fb-44c-432-843-fc3caefc1e8</td>
<td>10282</td>
</tr>
<tr>
<td>d300558f-7c1-4e77-5b58-65b7c6eab40</td>
<td>10240</td>
<td>d381bea-3768-4f12-a9b3-abf570ba918</td>
<td>10283</td>
</tr>
<tr>
<td>dd193eac-ddb4-444a-a5d-0f5cb15f1a5b</td>
<td>10241</td>
<td>8a3d3843-4ba9-4e9d-8578-9aa1887bb4</td>
<td>10284</td>
</tr>
</tbody>
</table>
| 7c06b072-b099-4d32-94a-c477307451b0 | 10242 | Continued on next page

19.4. Software Defined Networking 185
<table>
<thead>
<tr>
<th>Node UUID (compute_skylake rack 1)</th>
<th>Port</th>
<th>Node UUID (compute_skylake rack 2)</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>9779c17e-4e2a-4d85-7bd7-ec08aaf7f70d</td>
<td>10125</td>
<td>3e0c0e2c-04b7-4d29-9bba-2facc111ae</td>
<td>10157</td>
</tr>
<tr>
<td>9d478047-2ca8-4798-ac52-61686257f870</td>
<td>10126</td>
<td>149a9dbb-e312-4d7b-bc51-bf1a33329179</td>
<td>10158</td>
</tr>
<tr>
<td>afed1e2f-29f5-43b4-9dc6-5a5d3ca1e0ac</td>
<td>10127</td>
<td>80a084c0-4198-42fc-87ae-9fc1899eb336</td>
<td>10159</td>
</tr>
<tr>
<td>1a7ac31c-bb61-43e4-9350-b7390df58ae5</td>
<td>10128</td>
<td>b0194ee7-9866-4de1-a86d-2e6fda43c58a</td>
<td>10160</td>
</tr>
<tr>
<td>61ba06af-177b-4ea5-8e85-6519c0187c68</td>
<td>10129</td>
<td>4a351095-4f77-4a68-88c7-3a06b67b2269</td>
<td>10161</td>
</tr>
<tr>
<td>e46f61a-38f5-451d-a5ed-3b82a64444cb</td>
<td>10130</td>
<td>8c7b8067-cfa4-4994-9f12-77ce75631b8</td>
<td>10162</td>
</tr>
<tr>
<td>448de3a5-3200-4e6a-a4c3-e5348e570a0f</td>
<td>10131</td>
<td>219a586d-a3b2-4e4f-8517-ab72b8c2741</td>
<td>10163</td>
</tr>
<tr>
<td>21f7c8f2-b527-42a9-8f1e-c23cb6bdc91a</td>
<td>10132</td>
<td>03129b8e-330c-4591-bc17-96d7e153e74</td>
<td>10164</td>
</tr>
</tbody>
</table>

**CHI@TACC**

<table>
<thead>
<tr>
<th>Node UUID (compute_skylake rack 1)</th>
<th>Port</th>
<th>Node UUID (compute_skylake rack 2)</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>63aea74f-4e42-4f3e-b9b6-c5c473a911fa</td>
<td>10101</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c8b533e3-2576-4129-90cd-6456c859d98</td>
<td>10102</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8343783f-5429-405e-bc8f-dbfbbd83c158</td>
<td>10103</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9e14111e-2a0c-4cd1-a7c2-b2496a54600a</td>
<td>10104</td>
<td></td>
<td></td>
</tr>
<tr>
<td>110e303-3e3e-4da1-aaf6-a9e55236311a</td>
<td>10105</td>
<td></td>
<td></td>
</tr>
<tr>
<td>af9af518-b4c8-4b8f-a80c-a0181a471c8</td>
<td>10106</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6cccbf86-7722-49fd-b66b-46340f3add88</td>
<td>10107</td>
<td></td>
<td></td>
</tr>
<tr>
<td>f14b4bca-5db5-4285-874f-06c849e55f3</td>
<td>10108</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9fdb2370-cfd4-4396-baf7-7edade40c10d</td>
<td>10109</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4d49141f-4234-4731-88ca-57e0f0b5b6f</td>
<td>10110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7a0678ff-f111-e4c5-9198-2123207209be</td>
<td>10111</td>
<td></td>
<td></td>
</tr>
<tr>
<td>93b959f3-6d68-49cf-a322-4aacc516b46b</td>
<td>10112</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1bed206c-842c-4277-b2a3-cda4a8343628</td>
<td>10113</td>
<td></td>
<td></td>
</tr>
<tr>
<td>618ce3d2-4514-419d-b461-5935abcb0a7</td>
<td>10114</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9b4d3b1-13d9-4b60-950c-8685ad78008</td>
<td>10115</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b0fd7e7a-fec1-1d6e-9f41-94263a5b8267</td>
<td>10116</td>
<td></td>
<td></td>
</tr>
<tr>
<td>43d3723-8142-4b69-8a18-58f4e041c367</td>
<td>10117</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7571a78a-342f-41ef-81ed-cfdbe67674c8</td>
<td>10118</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fad519ed-a0b0-43d2-aec4-3c0ed2250814</td>
<td>10119</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1ee96615-2714-4166-82c5-b9b276e09186</td>
<td>10120</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8b60e599-fb22-4521-8b3a-a0e079a8f73</td>
<td>10121</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1985ed5-2c1e-444a-889d-e926b306f8d</td>
<td>10122</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9515fa6a-de4b-45b7-a98f-89d8b8831873</td>
<td>10123</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0b0b3eb9-14bf-423e-890f-3e187511d71</td>
<td>10124</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2a0aa343-ec4e-4683-4a7d-28df2d3c3d0</td>
<td>10125</td>
<td></td>
<td></td>
</tr>
<tr>
<td>43e67651-3d33-413e-8f77-19752d99803</td>
<td>10126</td>
<td></td>
<td></td>
</tr>
<tr>
<td>06b164d5-3514-4ebe-8928-0bd2950bb80</td>
<td>10127</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1e2e7689-527a-4608-9a1c-287ccf9b06a</td>
<td>10128</td>
<td></td>
<td></td>
</tr>
<tr>
<td>58c03a15-4421-4247-867c-4199a4f9e8b9</td>
<td>10129</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e5ab8372-050e-44af-95a4-cc3ed7963c87</td>
<td>10130</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9d058d81-05e5-441b-9462-1e17d86e1a6b</td>
<td>10131</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tf593140-57a0-4452-98dc-edb5b53f07e1</td>
<td>10132</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
19.4.3 Debugging Your Controller

To assist you in debugging your OpenFlow controller, Chameleon exposes a limited set of metrics exposed from the actual Corsa DP2000 switches. Currently you can see raw packet counts for each physical port on the switch. The data can be retrieved either from a Grafana web interface, or via a special Gnocchi metric resource accessed with the OpenStack CLI.

Accessing via Gnocchi Metrics

The metrics can also be queried via Gnocchi metrics. The metrics are located under specific Gnocchi resources and can be queried by passing the metric UUID.

```
# TACC Corsa #1
openstack --os-region CHI
@
TACC metric resource show 28596c49-0c14-5f08-a9e5-84790a05eef3
# UC Corsa #1
openstack --os-region CHI
@
UC metric resource show b72663e7-86fb-5785-82ed-b01ea9e0f282
# UC Corsa #2
openstack --os-region CHI
@
UC metric resource show 18e5e81e-798f-5299-9160-0f0ce34c17a9
```

This command will show all metrics available to view, and their UUID, e.g.:

```
+-----------------------+-------------------------------------------------------------
| Field | Value |
+-----------------------+-------------------------------------------------------------
| created_by_project_id | 4e9f3b6fbaf245e780b25fae2c166d4 |
| created_by_user_id | 5c9803db428c48daa2730892871a9242 |
| creator | 5c9803db428c48daa2730892871a9242:4e9f3b6fbaf245e780b25fae2c166d4 |
| ended_at | None |
| id | b72663e7-86fb-5785-82ed-b01ea9e0f282 |
| metrics | switch@if_rx_packets-1: c8144fb6-9a40-4eba-b3d4-516b1b9ddfff |
| | switch@if_rx_packets-2: 89947128-f794-4f01-bace-1b5a4fd93d32 |
| | ... |
| | switch@if_tx_packets-1: 4d66dea3-0d7a-4656-bd1e-813c6a4000d6 |
| | switch@if_tx_packets-2: 6dfc2627-03fe-4ce8-8497-1a40ccea60e3 |
| original_resource_id | collectd:chameleon-corsa1 |
| project_id | None |
```

(continues on next page)
You can then query an individual metric’s values with:

```
openstack metric measures show $METRIC_UUID
```

### 19.4.4 Corsa DP2000 Virtual Forwarding Contexts: Network Layout and Advanced Features

Virtual Forwarding Contexts (VFC) are the native OpenFlow abstraction used by the Corsa DP2000 series switches. Each VFC can be thought of as a virtual OpenFlow switch. Chameleon users can create VFCs by creating isolated networks on Chameleon via CLI or using complex appliances.

In this section, actual rack and switch layout of Skylake Nodes and Corsa DP2000 switches for both Chameleon sites is represented in the following figures. Also, example isolated networks with different controller options are shown along with associated VFCs and tunnels from Skylake Nodes are shown.

Users are able to specify an external OpenFlow controller and can assign a name to their VFCs. If an external controller is not specified, VFC is controlled by the OpenFlow controller (Learning Bridge Application) running on the switch.

1. Create an isolated network without an external OpenFlow controller and a VFC name:

   ```
   openstack network create --provider-network-type vlan --provider-physical-network physnet1 sdn-network-1
   ```

2. Create an isolated network with an external OpenFlow controller and without a VFC name:

   ```
   openstack network create --provider-network-type vlan --provider-physical-network physnet1 --description OFController=<OF_Controller_IP>:<OF_Controller_Port> sdn-network-2
   ```

3. Create an isolated network with an external OpenFlow controller and give a name to the VFC:

   ```
   openstack network create --provider-network-type vlan --provider-physical-network physnet1 --description OFController=<OF_Controller_IP>:<OF_Controller_Port>,VSwitchName=<VFCName> sdn-network-3
   ```

A named VFC will be created for the isolated network. Subsequent isolated networks that are created with the same VFC name specification will be attached to the same VFC. Current implementation lets the user specify only one OpenFlow controller to the VFCs. Also, subsequent isolated network creation commands should include exactly the same `--description`.
openstack network create --provider-network-type vlan --provider-physical-network physnet1 --description OFController=<OF_Controller_IP>:<OF_Controller_Port>,VSwitchName=<VFCName> sdn-network-4

4. Skylake Nodes at UC: **CHI@UC** has two racks with Skylake Nodes. Each rack has a TOR Corsa DP2000 series switch. VFCs for isolated networks are created on Corsa-1. Nodes on the second rack are connected to the VFC via statically provisioned VFCs on Corsa-2. You will see the ports on the VFCs as described in “Port Mapping” section.

5. Skylake Nodes at TACC: **CHI@TACC** has one rack with Skylake Nodes. You will see the ports on the VFCs as described in “Port Mapping” section.

### 19.4.5 Controllers for Corsa DP2000 series switches

OpenFlow controllers often need to be aware of the slight differences in implementation across switch vendors. What follows is a description of the quirks we have found while using the Corsa DP2000 series switches as well as a simple controller configuration that is compatible with Chameleon OpenFlow networks.

We have used Ryu and OpenDaylight controllers for the VFCs (Virtual Forwarding Context) on Corsa switches. We have provided a sample OpenFlow Ryu controller application that is available on GitHub. In addition, we have provided a Chameleon appliance that creates a Ryu controller based on these code modifications.

This controller is derived from the Ryu simple_switch_13.py with the following considerations. If you want use any other OpenFlow controller you will have to make similar considerations.

1. VFCs on Corsa switches are created by allocating specific amounts of system resources. Each VFC has a limited amount of resources in order to accommodate the requests of all Chameleon users. This limits the number of flows that can be put in the flow tables. Controllers will need to be careful not to fill up the flow tables. In our example, an idle timeout (defaulting to 5 minutes) to any rule inserted into the VFC via the controller is added to ensure the flow tables are cleaned up. This way, the switch removes the rule itself, once traffic matching the rule stops passing (for the specified interval).

2. The Corsa switches do not support Actions=FLOOD since this reserved port type is only for hybrid switches and it is optional. Corsa is an Openflow-only switch which supports the required port ALL. Controllers must replace the Actions=FLOOD to Actions=ALL in packet out messages.

3. Flow tables are modified according to the status of the ports being added or deleted from the VFC.

The following changes are made to the application:

Added the functions below:

```py
def _port_status_handler(self, ev):
    def delete_flow(self, datapath, port):
          
```

Added IDLE_TIMEOUT to flow modification in:

```py
def add_flow(self, datapath, priority, match, actions, buffer_id=None):
      
```

Changes are made in the function below to change Actions=FLOOD to actions=ALL in packet out message in the `def _packet_in_handler(self, ev):` method.

This controller application can be run by the script below:
19.4. Software Defined Networking
19.5 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 please 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.

19.5.1 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.

```
opestack 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 enol`. The first Ethernet interface is typically `eno1` for most Chameleon base images.

19.5.2 Enabling Jumbo Frames on Existing Network

You can also modify the MTU of an existing network using the command below. Please note that this will only effect newly created baremetal instances.

```
openstack network set <network_name> --mtu 9000
```

19.5.3 Enabling Jumbo Frames on Existing Instances

Setting the MTU on your Chameleon network only effects 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 enol mtu 9000
```
20.1 Introduction

Chameleon provides access to five FPGA nodes. Four nodes are located at CHI@TACC. Each of these nodes is fitted with a Nallatech 385A board with an Altera Arria 10 1150 GX FPGA (up to 1.5 TFlops), 8 GB DDR3 on-card memory, and dual QSFP 10/40 GbE support. One node is located at CHI@UC. The node is fitted with a Terasic DE5a-Net board with an Altera Arria 10 GX 1150 FPGA (up to 1.5 TFlops), 4 GB DDR3 on-card memory, and four QSFP 10/40 GbE support. All FPGA nodes are configured to run OpenCL code, but they can be reconfigured (by a request to our Help Desk) to run compiled designs prepared with Altera Quartus.

Due to export control limitations, access to the development toolchain requires verification of your user profile. This guide explains how to gain access to the development toolchain and execute code on the FPGA nodes. Briefly, the steps for building an FPGA application are:

- Create a TACC account at the TACC Portal
- Setup Multi-Factor Authentication for TACC Resources by following this documentation
- Request access to the FPGA Build Node project at the Help Desk
- SSH to the fpga01.tacc.chameleoncloud.org host to build your FPGA application
- Use scp to copy your FPGA application from fpga01.tacc.chameleoncloud.org to the FPGA node you wish to run it on

20.2 Development

Chameleon provides a build system that includes the necessary Altera SDK for OpenCL tools for developing kernels for use on the Nallatech 385A cards and the Terasic DE5a-Net card, both using the Altera Arria 10 FPGA.

Due to licensing requirements, you must apply for access to the FPGA build system. Submit a ticket through our help system to request access.

FPGA resources are only available at CHI@TACC. Due to TACC’s security requirements, multi-factor authentication must be used to access the FPGA build system. You can either use a smartphone app (Apple iOS or Android) or SMS
messaging: follow this documentation to set it up. Once you have set up multi-factor authentication, you can SSH to fpga01.tacc.chameleoncloud.org with your Chameleon username and password; you will also be asked for a TACC security token, which will be provided to you via the app or SMS.

Each user’s home directory will contain an archive file containing a Hello World OpenCL example: `exm_opencl_hello_world_x64_linux_16.0.tgz`. Extract the archive with the following command:

```
tar -zxf exm_opencl_hello_world_x64_linux_16.0.tgz
```

Two directories will be extracted: `common` and `hello_world`. Change into the `hello_world` directory.

```
cd hello_world
```

Prior to compiling, load the Quartus environment configuration for either the Nallatech or Terasic board.

**Nallatech:**

```
module load nallatech
```

**Terasic:**

```
module load terasic
```

**Important:**

The host code contains the function `findPlatform(Altera)`, which searches for the “Altera” platform name. It should instead be instructed to search for “Intel(R) FPGA”. This change can be made by editing `../hello_world/host/src/main.cpp`:

```
findPlatform("Intel(R) FPGA")
```

Compiling an OpenCL kernel often takes a very long time, so it is essential to debug by using the emulation feature of the compiler using `-march=emulator` in the compiler command. Note that the `--board p385a_sch_ax115` parameter is required for the Nallatech board, and the `-board=de5a_net_e1` parameter is required for the Terasic board. These correctly identify the FPGA boards available on Chameleon. Do not alter these parameters or their syntax. In this example, the host application requires the output name to be `hello_world.aocx`, so this parameter must also be unchanged.

**Nallatech:**

```
aoc --board p385a_sch_ax115 device/hello_world.cl -o bin/hello_world.aocx -march=emulator
```

**Terasic:**

```
aoc -board=de5a_net_e1 device/hello_world.cl -o bin/hello_world.aocx -march=emulator
```

Build the host application, which is used to execute the OpenCL kernel.

```
make
```

Now run the emulated kernel.

```
env CL_CONTEXT_EMULATOR_DEVICE_INTELFPGA=1 ./bin/host
```

When debugging is complete and the code is ready to be compiled for the FPGA hardware, remove the emulation flag. This may take several hours to complete, so we recommend you run it inside a terminal multiplexer, such as screen or tmux which are both installed on the build node.

**Nallatech:**

```
```
aoc --board p385a_sch_ax115 device/hello_world.cl -o bin/hello_world.aocx

Terasic:

aoc -board=de5a_net_e1 device/hello_world.cl -o bin/hello_world.aocx

## 20.3 Execution

After completing development of an OpenCL kernel on our build node, the kernel and host application must be transferred and executed on a node with an FPGA accelerator.

When using CHI@TACC GUI to reserve nodes, use the **Node Type to Reserve** selector and choose **FPGA**. Alternatively, use the Resource Discovery web interface to reserve a node equipped with an FPGA accelerator card by filtering the node selection using the **with FPGA** button, and clicking **Reserve** at the bottom of the selection. Copy the generated CLI command and use it to create your reservation.

In order to have access to the required runtime environment for using the FPGAs, use the image **CC-CentOS7-FPGA** when launching your instance.

Log in to the instance, download the application code (both **common** and **hello_world** directories) from the build system using **scp**, and change into the **hello_world** directory:

```
scp -r <username>@fpga01.tacc.chameleoncloud.org:~/common .
scp -r <username>@fpga01.tacc.chameleoncloud.org:~/hello_world .
```

```
cd hello_world
```

Compile the host application, if necessary.

```
make
```

Program FPGA with the OpenCL kernel, using **acl0** as the device name.

```
aocl program acl0 ./bin/hello_world.aocx
```

**Attention:** If you are at CHI@UC, please run the following commands (program FPGA as **root**).

```
sudo -i
source /etc/profile.d/altera.sh
cd /home/cc/hello_world
aocl program acl0 ./bin/hello_world.aocx
```

Execute the host application to run on FPGA.

```
./bin/host
```

You should see an output like the following:

```
Querying platform for info:
==========================
CL_PLATFORM_NAME = Altera SDK for OpenCL
CL_PLATFORM_VENDOR = Altera Corporation
CL_PLATFORM_VERSION = OpenCL 1.0 Altera SDK for OpenCL, Version
˓→16.0
```

(continues on next page)
Querying device for info:

```
CL_DEVICE_NAME = p385a_sch_ax115 : nalla_pcie (aclnalla_
   → pcie0)
CL_DEVICE_VENDOR = Nallatech ltd
CL_DEVICE_VENDOR_ID = 4466
CL_DEVICE_VERSION = OpenCL 1.0 Altera SDK for OpenCL, Version
   → 16.0
CL_DRIVER_VERSION = 16.0
CL_DEVICE_ADDRESS_BITS = 64
CL_DEVICE_AVAILABLE = true
CL_DEVICE_ENDIAN_LITTLE = true
CL_DEVICE_GLOBAL_MEM_CACHE_SIZE = 32768
CL_DEVICE_GLOBAL_MEM_CACHELINE_SIZE = 0
CL_DEVICE_GLOBAL_MEM_SIZE = 8589934592
CL_DEVICE_IMAGE_SUPPORT = true
CL_DEVICE_LOCAL_MEM_SIZE = 16384
CL_DEVICE_MAX_CLOCK_FREQUENCY = 1000
CL_DEVICE_MAX_COMPUTE_UNITS = 1
CL_DEVICE_MAX_CONSTANT_ARGS = 8
CL_DEVICE_MAX_CONSTANT_BUFFER_SIZE = 2147483648
CL_DEVICE_MAX_WORK_ITEM_DIMENSIONS = 3
CL_DEVICE_MEM_BASE_ADDR_ALIGN = 8192
CL_DEVICE_MIN_DATA_TYPE_ALIGN_SIZE = 1024
CL_DEVICE_PREFERRED_VECTOR_WIDTH_CHAR = 4
CL_DEVICE_PREFERRED_VECTOR_WIDTH_SHORT = 2
CL_DEVICE_PREFERRED_VECTOR_WIDTH_INT = 1
CL_DEVICE_PREFERRED_VECTOR_WIDTH_LONG = 1
CL_DEVICE_PREFERRED_VECTOR_WIDTH_FLOAT = 1
CL_DEVICE_PREFERRED_VECTOR_WIDTH_DOUBLE = 0
Command queue out of order? = false
Command queue profiling enabled? = true
Using AOCX: hello_world.aocx
Reprogramming device with handle 1

Kernel initialization is complete.
Launching the kernel...

Thread #2: Hello from Altera's OpenCL Compiler!

Kernel execution is complete.
CHAPTER 21

Experiment Precis

21.1 Introduction

Chameleon records experiment setup (OpenStack) events that users performed on the testbed, such as creating leases, creating instances, and setting up networks. Users can request their experiment records from Chameleon using their Chameleon credentials. A report on those experiment records is known as the Experiment Precis. An Experiment Precis is bounded to a lease. Chameleon defines an experiment as a series of testbed setup (OpenStack) events a user performed under a lease of a project. Using Experiment Precis, users will be able to analyze, understand and even replay their experiments.

Currently, Experiment Precis is provided in JSON format. The following contents are included:

```json
{
    "experiment_precis_name": "The name of the experiment precis",
    "experiment_precis_id": "The id of the experiment precis",
    "site": "Which Chameleon site the experiment was performed",
    "testbed_version": "The testbed version at the time the experiment started",
    "report_time_in_utc": "The datetime when the experiment precis was requested (in UTC)",
    "report_time_in_ct": "The datetime when the experiment precis was requested (in CT)",
    "events": "A list of events performed",
    "event_page": "The page number of the events",
    "event_page_size": "The page size of the events",
    "hardware": "Hardwares that were used during the experiment",
    "hardware_page": "The page number of the hardwares",
    "hardware_page_size": "The page size of the hardwares",
    "metric": "Metrics saved in Gnocchi",
    "metric_page": "The page number of the metrics",
    "metric_page_size": "The page size of the metrics"
}
```
21.2 Installation

To request your experiment precis from Chameleon, you need to install the Chameleon Experiment Precis (cep) Client. To install cepclient on your local machine, run the following command:

```
pip install cepclient
```

To test if cepclient is properly installed, run:

```
cep --help
```

21.3 Setting up cep client

Before using the cep client, you must configure the authentication-related environment variables via `source the OpenStack RC script` or provide the authentication values as command parameters. If you choose to pass command parameters, use `cep --help` and look for Authentication Options for more information.

**Note:** When using rc script for setting up cep client, please download and use the v3 file.

21.4 List experiment precis

You can use `list` command to find all the experiments you have run.

```
cep list
```

The output looks like the following:

```
+---------------------+---------------------+--------------------------------------+--
| created_at | updated_at | id | name | lease_id |
+---------------------+---------------------+--------------------------------------+--
| 2018-10-18 09:21:02 | 2018-10-18 09:21:02 | 0fa88391-6b14-465d-a62c-91ad8f6eb920 | 0fa88391-6b14-465d-a62c-91ad8f6eb920 | 972b70aa-33ca-42fc-9d4e-e07b2b9df3c3 |
| 2018-10-18 10:05:50 | 2018-10-18 10:05:50 | 93ffbb79-e732-4046-a49d-b223ff8f1bd5 | 93ffbb79-e732-4046-a49d-b223ff8f1bd5 | 9f91c7ac-212b-4d46-8f88-1e9db341f41a |
+---------------------+---------------------+--------------------------------------+--
```

The Experiment Precis will be listed in the reverse order of creation datetime, i.e. the latest Experiment Precis is listed the first.

For more information, run:

```
cep list --help
```
21.5 Rename experiment precis

Initially, Chameleon sets the name of an *Experiment Precis* the same as its id. However, you can rename it for the convenience of future retrieving. To rename an *Experiment Precis*, run the following command:

```
cep rename --name <new_name> <ep_id or ep_name>
```

**Tip:** Renaming your experiment precis to a meaningful name will help you 1) mark your *special* experiment; 2) understand what the experiment is about; 3) retrieve your experiment precis.

For more information, run:

```
cep rename --help
```

21.6 Print experiment precis

Finally, you can retrieve all the details about your experiment by using the `print` command.

```
cep print <ep_id or ep_name>
```

The above command will print the requested experiment precis on your terminal in a compact format. To pretty-print the experiment precis, add `--pretty` to the command. To print the experiment precis to a file, add `--output <path/to/file>` to the command.

The following is an example of `cep print` output:

```json
{
  "event_page": 0,
  "event_page_size": -1,
  "events": [
    {
      "event_time": "2018-10-18 15:05:50",
      "event_type": "lease.create",
      "metadata": {
        "end_date": "2018-10-19T15:05:00.000000",
        "start_date": "2018-10-18T15:06:00.000000"
      },
      "resource_id": "9f91c7ac-212b-4d46-8f88-1e9db341f41a",
      "service": "blazar"
    },
    {
      "event_time": "2018-10-18 15:06:05",
      "event_type": "lease.event.start_lease",
      "metadata": {
        "end_date": "2018-10-19T15:05:00.000000",
        "start_date": "2018-10-18T15:06:00.000000"
      },
      "resource_id": "9f91c7ac-212b-4d46-8f88-1e9db341f41a",
      "service": "blazar"
    },
    ...
  }
}
(continues on next page)
{"event_time": "2018-10-19 15:05:11",
"event_type": "lease.event.end_lease",
"metadata": {
   "end_date": "2018-10-19T15:05:00.000000",
   "start_date": "2018-10-18T15:06:00.000000"
},
"resource_id": "9f91c7ac-212b-4d46-8f88-1e9db341f41a",
"service": "blazar"
}

"experiment_precis_id": "93ffbb79-e732-4046-a49d-b223ff8f1bd5",
"experiment_precis_name": "zhenz-test-2",

"hardware": {
    "architecture": {
       "platform_type": "x86_64",
       "smp_size": 2,
       "smt_size": 48
    },
    "bios": {
       "release_date": "03/09/2015",
       "vendor": "Dell Inc.",
       "version": 1.2
    },
    "chassis": {
       "manufacturer": "Dell Inc.",
       "name": "PowerEdge R630",
       "serial": "8Q28C42"
    },
    "gpu": {
       "gpu": false
    },
    "links": [\
               {"href": "/sites/uc/clusters/chameleon/nodes/b0525159-5c95-4b71-83f2-b8d6bdd2acd2",
                "rel": "self",
                "type": "application/vnd.grid5000.item+json"
               },\
               {"href": "/sites/uc/clusters/chameleon",
                "rel": "parent",
                "type": "application/vnd.grid5000.item+json"
               },\
               {"href": "/sites/uc/clusters/chameleon/nodes/b0525159-5c95-4b71-83f2-b8d6bdd2acd2/versions/53c90ef0512d5013ee30d431cd62e68bdf34d4ca",
                "rel": "version",
                "type": "application/vnd.grid5000.item+json"
               },\
               {"href": "/sites/uc/clusters/chameleon/nodes/b0525159-5c95-4b71-83f2-b8d6bdd2acd2/versions",
                "rel": "versions",
                "type": "application/vnd.grid5000.collection+json"
               }]
"main_memory": {
  "humanized_ram_size": "128 GiB",
  "ram_size": 134956859392
},
"monitoring": {
  "wattmeter": false
},
"network_adapters": [
  {
    "bridged": false,
    "device": "eno1",
    "driver": "bnx2x",
    "interface": "Ethernet",
    "mac": "44:a8:42:15:c4:dd",
    "management": false,
    "model": "NetXtreme II BCM57800 1/10 Gigabit Ethernet",
    "mounted": true,
    "rate": 100000000000,
    "vendor": "Broadcom Corporation"
  },
  {
    "bridged": false,
    "device": "eno2",
    "driver": "bnx2x",
    "interface": "Ethernet",
    "mac": "44:a8:42:15:c4:df",
    "management": false,
    "model": "NetXtreme II BCM57800 1/10 Gigabit Ethernet",
    "mounted": false,
    "rate": 100000000000,
    "vendor": "Broadcom Corporation"
  },
  {
    "bridged": false,
    "device": "eno3",
    "driver": "bnx2x",
    "interface": "Ethernet",
    "mac": "44:a8:42:15:c4:e1",
    "management": false,
    "model": "NetXtreme II BCM57800 1/10 Gigabit Ethernet",
    "mounted": false,
    "rate": 100000000000,
    "vendor": "Broadcom Corporation"
  },
  {
    "bridged": false,
    "device": "eno4",
    "driver": "bnx2x",
    "interface": "Ethernet",
    "mac": "44:a8:42:15:c4:e3",
    "management": false,
    "model": "NetXtreme II BCM57800 1/10 Gigabit Ethernet",
    "mounted": false,
    "rate": 100000000000,
    "vendor": "Broadcom Corporation"
}]

21.6. Print experiment precis 203
"node_type": "compute_haswell",
"placement": {
    "node": 14,
    "rack": 1
},
"processor": {
    "cache_l1": null,
    "cache_l1d": 32768,
    "cache_l1i": 32768,
    "cache_l2": 262144,
    "cache_l3": 31457280,
    "clock_speed": 3100000000,
    "instruction_set": "x86-64",
    "model": "Intel Xeon",
    "other_description": "Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz",
    "vendor": "Intel",
    "version": "E5-2670 v3"
},
"storage_devices": [
    {
        "device": "sda",
        "driver": "megaraid_sas",
        "humanized_size": "250 GB",
        "interface": "SATA",
        "model": "ST9250610NS",
        "rev": "AA63",
        "size": 250059350016,
        "vendor": "Seagate"
    }
],
"supported_job_types": {
    "besteffort": false,
    "deploy": true,
    "virtual": "ivt"
},
"type": "node",
"uid": "b0525159-5c95-4b71-83f2-b8d6bdd2acd2",
"version": "53c90ef0512d5013ee30d431cd62e68bfd34d4ca"
],
"hardware_page": 0,
"hardware_page_size": 25,
"metric_page": 0,
"metric_page_size": 25,
"metrics": [
    {
        "instance_id": "44ad06ee-41d7-48f9-a52a-179030754707",
        "metric_id": "dd22e02386714516a913d96659617eb",
        "metric_name": "interface-enol@if_dropped"
    },
    {
        "instance_id": "44ad06ee-41d7-48f9-a52a-179030754707",
        "metric_id": "512ac94754b64906a12960d1f0a929c9",
        "metric_name": "interface-enol@if_errors"
    }
],
The events section is a list of testbed events ordered by event timestamp. The hardware section contains information of all the nodes that were used in the experiment. The hardware information is retrieved by using the same method as the Resource Discovery. The metrics section is a list of the metrics captured during the experiment. The Experiment Precis only contains the instance_id, metric_id, and metric_name in the metrics list. You can use the openstack metric command line to get all the measurements of a particular metric over time for an instance. For more information, run:

```
cep print --help
```

**Important:** Chameleon only keeps an experiment precis for 180 days. Please make sure to save your experiment precis you’d like to keep for a longer time by using cep print command. You can output it to a file and keep it as a record.

### 21.6.1 Pagination

In the case of “large” experiment with large number of nodes and metrics, events, hardwares, and metrics are printed in pages. By default, the page number is set to 0 and the page size is set to 25. However, you can tune the pagination by specifying the following parameters:

```
--event-page-size EVENT_PAGE_SIZE
  Page size for event; ignored if event is excluded; set to negative value to show all
--event-page EVENT_PAGE
  Page number for event; ignored if event is excluded
--metric-page-size METRIC_PAGE_SIZE
  Page size for metric; ignored if metric is excluded; set to negative value to show all
--metric-page METRIC_PAGE
  Page number for metric; ignored if metric is excluded
--hardware-page-size HARDWARE_PAGE_SIZE
  Page size for hardware; ignored if hardware is excluded; set to negative value to show all
--hardware-page HARDWARE_PAGE
  Page number for hardware; ignored if hardware is excluded
```

**Tip:** To show all, set page size to a negative value. If page size is negative, page parameter will be ignored. Negative
21.6.2 Filters

The `cep` tool provides multiple filters to help you focus on the contents you care.

**Event Filters**

- To exclude all the events from the *Experiment Precis*, use `--exclude-event`.
- To include or exclude services, use `--include-services` and/or `--exclude-services`. For example, if you only want to print `blazar` (reservation) and `nova` (instance) related events, run the following command:

  ```
  cep print --pretty --include-services blazar,nova <ep_id or ep_name>
  ```

- You can exclude event metadata by passing `--exclude-event-metadata`.
- You can apply datetime filters to your events. For example, to print events up to `2018-10-05 00:00:00`, run:

  ```
  cep print --pretty --end-datetime "2018-10-05 00:00:00" <ep_id or ep_name>
  ```

  Or to print events from `2018-10-05 09:00:00` to `2018-10-05 17:00:00`, run:

  ```
  cep print --pretty --start-datetime "2018-10-05 09:00:00" --end-datetime "2018-10-05 17:00:00" <ep_id or ep_name>
  ```

**Note:** When using datetime filters, use datetime in UTC.

**Metric Filters**

- To exclude metrics from the *Experiment Precis*, use `--exclude-metric`.

**Hardware Filters**

- To exclude hardware information from the *Experiment Precis*, use `--exclude-hardware`.
Chameleon Trovi is a sharing portal that allows you to share digital research and education artifacts, such as 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.

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.

### 22.1 Browsing artifacts

Trovi allows you to browse artifacts, presented in a scrolling list format. On the right hand side, there are multiple filtering options. By clicking on the project name, you can see all the artifacts that belong to that project. The “All” choice shows you all of the artifacts you have access to. You can also see how many times people have downloaded and launched your notebook with the icons in the bottom left corner of an artifact.

**Note:** Some Trovi artifacts are supported by the Chameleon team and are denoted with a small Chameleon logo. You can contact the Help Desk if you are using these artifacts and encounter issues.

### 22.1.1 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, please contact the Help Desk if you need more space). The animation below shows how easy it is:
A configurable, experimemt environment for large-cloud research.

Fig. 1: The “Trovi” option under the “Experiment” section takes you to Trovi.

Fig. 2: Clicking the “Launch with JupyterHub” button to import a Trovi artifact into your own Jupyter server.
22.2 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.

22.2.1 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.

22.2.2 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.

22.2.3 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 wish 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.

4. **Share with projects**: this option allows you to share the experiment with members of a Chameleon Project. You can also share amongst multiple projects.
To make your artifact shareable follow the actions shown in the animation below: first select how you want to share and then your sharing destination.

**Publishing to Zenodo**

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.

This also creates a DOI, which you can easily include in your paper. The artifacts shared on Zenodo also appear on Trovi.

**22.3 Using Daypass**

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 tied to your 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 the 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.

**22.3.1 Allowing Reproducibility Requests**

First, the owner of an artifact 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, an allocation request is automatically made. 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 in an active Chameleon project.

**Requesting a Daypass**

When you select “Request Daypass”, you will be taken a form where you are asked to submit your name, institution, and a reason why you want to reproduce the artifact. The goal of this form is to explain to the PI why you are interested
A person without an active Chameleon allocation is unable to launch and reproduce your artifact. If you enable reproducibility requests, users can send requests, and if granted, they will be given a small temporary allocation on a separate project. While usage by reproducing users will not count towards your project's main allocation, the PI is nonetheless accountable for their responsible usage of Chameleon resources. All reproducibility requests therefore require PI approval.

- Enable reproducibility requests

**Hours a user has to reproduce**

Hours a user has to reproduce

---

in reproducing the artifact, as the PI is still responsible for overseeing the proper use of Chameleon resources, and the PI may be a stranger to you.

---

After submitting the form, the managers (and PI) of the project associated with the artifact will receive an email informing them of the request.

---

**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.

**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. Please note the ID of the project (CHI-XXXXX), which may be needed to configure an artifact.

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.
23.1 Introduction

OpenStack is an Infrastructure as a Service (IaaS) platform that allows you to create and manage virtual environments. Chameleon provides an installation of OpenStack Train 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.

This documentation provides 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 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.

23.2 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 web interface 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.

23.2.1 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.
23.2.2 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.

Tip: If you select different flavors from the Flavor dropdown, their characteristics are displayed on the right.

When you are finished with this step, go to the Key Pair Tab.

Select an SSH keypair that will be inserted into your virtual machine. You will need to select a keypair 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.

### 23.2.3 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.

### 23.2.4 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.
23.2.5 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.

**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.
23.2. Work with KVM using the GUI
Create Security Group

Name
newgroup

Description:
Security groups are sets of IP filter rules that are applied to network interfaces of a VM. After the security group is created, you can add rules to the security group.

Security Groups

<table>
<thead>
<tr>
<th>Name</th>
<th>Security Group ID</th>
<th>Description</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allow HTTP</td>
<td>4a391e7f-650a-445a-9d21-41a1a5d0655f</td>
<td></td>
<td>Manage Rules</td>
</tr>
<tr>
<td>Allow SSH</td>
<td>1b335a5a-9b89-4e02-bb55-46735f13601d</td>
<td></td>
<td>Manage Rules</td>
</tr>
<tr>
<td>default</td>
<td>e93220b6-f163-49b5-861d-95f8e8669f1</td>
<td>Default security group</td>
<td>Manage Rules</td>
</tr>
<tr>
<td>newgroup</td>
<td>69a52b3a-4c81-4894-9b2a-2033ac692d3f</td>
<td></td>
<td>Manage Rules</td>
</tr>
<tr>
<td>vacode</td>
<td>6a0b2108-4a77-4b9a-9a08-5c367a3ac3f</td>
<td></td>
<td>Manage Rules</td>
</tr>
</tbody>
</table>

Manage Security Group Rules: newgroup (09e352a5-c821-4b44-ba22-e3af0001ce45)

<table>
<thead>
<tr>
<th>Direction</th>
<th>Other Type</th>
<th>IP Protocol</th>
<th>Port Range</th>
<th>Remote IP Prefixes</th>
<th>Remote Security Group</th>
<th>Description</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Egress</td>
<td>IPv4</td>
<td>Any</td>
<td>Any</td>
<td>0.0.0.0</td>
<td>-</td>
<td>-</td>
<td>Delete Rule</td>
</tr>
<tr>
<td>Egress</td>
<td>IPv6</td>
<td>Any</td>
<td>Any</td>
<td>:0</td>
<td>-</td>
<td>-</td>
<td>Delete Rule</td>
</tr>
</tbody>
</table>
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.
23.2.6 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.

![Edit Security Groups dialog]

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.

23.2.7 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 connections. 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 balancing. 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 services 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 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 associating a Floating IP to a virtual machine.
23.2. Work with KVM using the GUI
Create Load Balancer

Add members to the load balancer pool.

**Allocated Members**

<table>
<thead>
<tr>
<th>IP Address</th>
<th>Subnet</th>
<th>Port</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

No members have been allocated

Add external member

**Available Instances**

<table>
<thead>
<tr>
<th>Name</th>
<th>IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>http</td>
<td></td>
</tr>
<tr>
<td>http-1</td>
<td>10.71.52.144</td>
</tr>
<tr>
<td>http-2</td>
<td>10.71.52.82</td>
</tr>
<tr>
<td>http-3</td>
<td>10.71.52.245</td>
</tr>
</tbody>
</table>

Add

Create Load Balancer

Add members to the load balancer pool.

**Allocated Members**

<table>
<thead>
<tr>
<th>IP Address</th>
<th>Subnet</th>
<th>Port</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.71.52.144</td>
<td>Chameleon Ops-subnet</td>
<td>80</td>
<td></td>
</tr>
</tbody>
</table>

Remove

Add external member

**Available Instances**

<table>
<thead>
<tr>
<th>Name</th>
<th>IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>http</td>
<td></td>
</tr>
<tr>
<td>http-1</td>
<td>10.71.52.144</td>
</tr>
<tr>
<td>http-2</td>
<td>10.71.52.82</td>
</tr>
<tr>
<td>http-3</td>
<td>10.71.52.245</td>
</tr>
</tbody>
</table>

Add

Cancel

< Back Next >

Create Load Balancer

Add members to the load balancer pool.

**Allocated Members**

<table>
<thead>
<tr>
<th>IP Address</th>
<th>Subnet</th>
<th>Port</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

No members have been allocated

Add external member

**Available Instances**

<table>
<thead>
<tr>
<th>Name</th>
<th>IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>http</td>
<td></td>
</tr>
<tr>
<td>http-1</td>
<td>10.71.52.144</td>
</tr>
<tr>
<td>http-2</td>
<td>10.71.52.82</td>
</tr>
<tr>
<td>http-3</td>
<td>10.71.52.245</td>
</tr>
</tbody>
</table>

Add
instance. Making changes to the various components of the load balancer by clicking on the blue-colored 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.

## 23.3 Work with KVM using the CLI

For general information on CLI authentication and use, please see the command-line-interface section.

### 23.3.1 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:

```bash
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:

```bash
qemu-img convert -f qcow2 -O raw <original.qcow2> <converted.img>
```

Once converted, use glance to upload the image:

```bash
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.