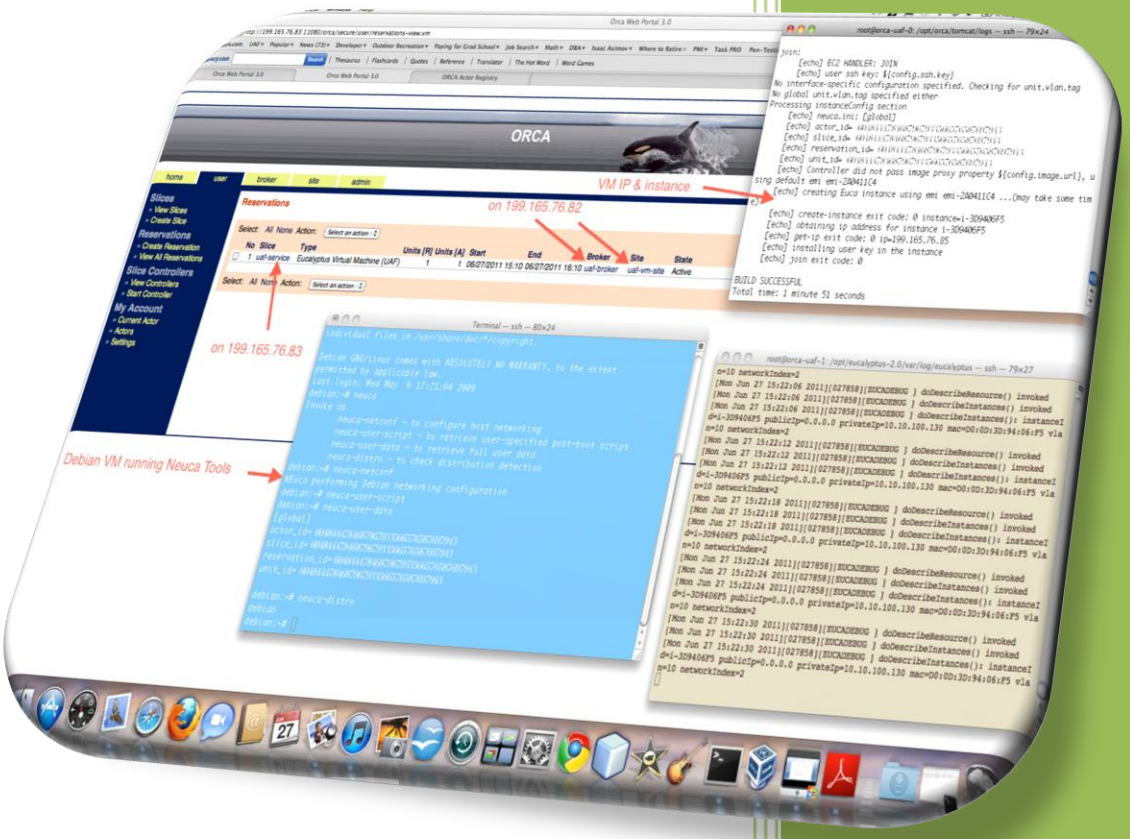
The Global Environment for Network Innovation

GENI 2011

Open Resource Control Architecture (ORCA) User's Manual



Compiled by John Quan

Computer Science Department

University of Alaska Fairbanks

6/3/2011

Table of Contents

Introduction to ORCA

Read this introduction by the Renaissance Computing Institute (RENCI):

https://geni-orca.renci.org/trac/wiki/orca-introduction

Purpose

ORCA is still in development as of this writing, and so much of how one interacts with it changes with each new release. For this reason, the purpose of this manual is to give a basic explanation of UAF's ORCA cluster set up and to demonstrate how one can create Eucalyptus virtual machine instances. This manual does not provide any direction for ORCA handler, package, or plug-in functionality, but may do so in the future as ORCA standards develop.

System Setup

UAF ORCA consists of two virtual technology (VT) enabled servers with two Network Interface Cards (NIC) each, a Cisco 2950 programmable switch, and a "dumb" switch. The servers run the Ubuntu[1] 11.04 Natty Narwhal operating system, with Image Proxy[2] installed on the Head Node, and ORCA[3] Camano 3.0 and Eucalyptus[4] 2.0.2 installed on both the Head and Worker nodes. In a production environment, Image Proxy, ORCA, and Eucalyptus might reside on hundreds of servers—think Amazon Cloud scale—but my goal was to set up a UAF ORCA cluster using as few resources as possible to achieve compactness and lower costs.

Table 1. Basic configuration

Head Node	
ORCA Web Portal	199.165.76.82:11080/orca
uaf-vm-site	
uaf-broker	
Eucalyptus	
Walrus	
Cloud Controller	
Cluster Controller	
Image Proxy	199.165.76.82:11081/repository/services
Hosts a Neuca-enabled Debian 5.0 image	
Worker Node	
ORCA Web Portal	199.165.76.83:11080/orca
uaf-service	
Eucalyptus	
Node Controller	
Eucalyptus instances (virtual machines)	199.165.76.84-94

Image Proxy

ORCA relies on Image Proxy to distribute images, from which Eucalyptus creates virtual machines (VM). It utilizes an axis2 server to host images from any URL and thus makes the images available to other ORCA clusters. For instance, Duke University can use our hosted images and vice versa. In our case, the Head node hosts a Debian 5.0 image at port 11081 for compactness, but one could dedicate one or more servers to host VM images. If Eucalyptus cannot connect to an Image Proxy server, then it uses the default image on the Eucalyptus cluster controller.

Head and Worker Nodes

The Head node hosts ORCA and Eucalyptus. The ORCA container does not have to reside on the Head or Worker to control the Eucalyptus cluster, but the Head node hosts the actors uaf-vm-site and uaf-broker

[1] www.ubuntu.com
[2] https://code.renci.org/gf/project/networkedclouds/wiki/?pagename=ImageProxy
[3] https://geni-orca.renci.org/trac/wiki/
[4] http://www.eucalyptus.com

for compactness. Currently, ORCA recommends that the slice manager—on the Worker—remain separate from the other actors, which is one reason why an ORCA cluster requires at least two servers.

In addition, the Head node hosts the Eucalyptus Walrus, Cloud Controller, and Cluster Controller (CC). These entities have a one-to-many relationship, with one Walrus having many Cloud Controllers, one Cloud Controller having many CCs, and one CC having many Node Controllers (NC)—one NC is on the Worker. The CC controls all networking by allocating public IP addresses (199.165.76.84-94) through Dynamic Host Configuration Protocol (DHCP). Slice Managers, such as uaf-service, can then give out the private key to users, who access the VM using a Secure Shell (SSH) connection. Users are actually connecting to the CC, which converts the public IP to a private IP to pass communications and manage the NC.

The Worker node hosts ORCA and Eucalyptus, as well. The major differences between the Head and Worker are that the Worker does not host Image Proxy (but it could), it only has the actor uaf-service, it only has a NC, and it actually runs the Eucalyptus instances—the VMs. Currently, Eucalyptus recommends that the NC remain separate from the other Eucalyptus entities, which is another reason why an ORCA cluster requires at least two servers.

The Actor Registry

The Actor Registry at https://geni.renci.org:11443/registry/actors.jsp lists all approved actors in ORCA. One can share resources with other actors by joining geni-orca-users@googlegroups.com and contacting them directly.

Yes Manage	uaf-broker	cd72228b-130e-4e97-ae3f-6ed117b13ef4	ORCA Broker	UAF Broker	Link	orca.shirako.core.Broker	orca.policy.core.BrokerSimplerUnitsPolicy	Click for Public Key	Click for Actor Certificate	N/A	N/A
Yes Manage	duke-service1	59a1fe71-4fe9-4075-b231-b95b9c116549	ORCA Service Manager (SM)	Duke Service Manager 1	Link	orca.shirako.core.ServiceManager	orca.policy.core.ServiceManagerSimplePolicy	Click for Public Key	Click for Actor Certificate	N/A	N/A
Yes Manage	ndl-broker	25bc9111-9b41-46ab-a96b-3c87f574cfde	ORCA Broker	NDL enabled broker hosted at RENCI	Link	orca.shirako.core.Broker	orca.policy.core.BrokerSimplerUnitsPolicy	Click for Public Key	Click for Actor Certificate	N/A	N/A
Yes Manage	uaf-service	1d5616cc-4751-458d-acfa-c36719fdf7ef	ORCA Service Manager (SM)	UAF Service Manager	Link	orca.shirako.core.ServiceManager	orca.policy.core.ServiceManagerSimplePolicy	Click for Public Key	Click for Actor Certificate	N/A	N/A
Yes Manage	uaf-vm-site	39c74217-115b-4f5f-9d42-f87510aba462	ORCA Site Authority / Aggregate Manager (AM)	UAF Euca site authority	Link	orca.shirako.core.Authority	orca.policy.core.AuthorityCalendarPolicy	Click for Public Key	Click for Actor Certificate	Click for Abstract Site NDL	Click for Full Site NDL
			ORCA Site				orca.policy	Click			

Log In

Log in to ORCA by opening http://199.165.76.82:11080/orca/ in a web browser and accepting the RENCI self-signed certificate. Once logged in, five tabs appear: home, user, broker, site, and admin.

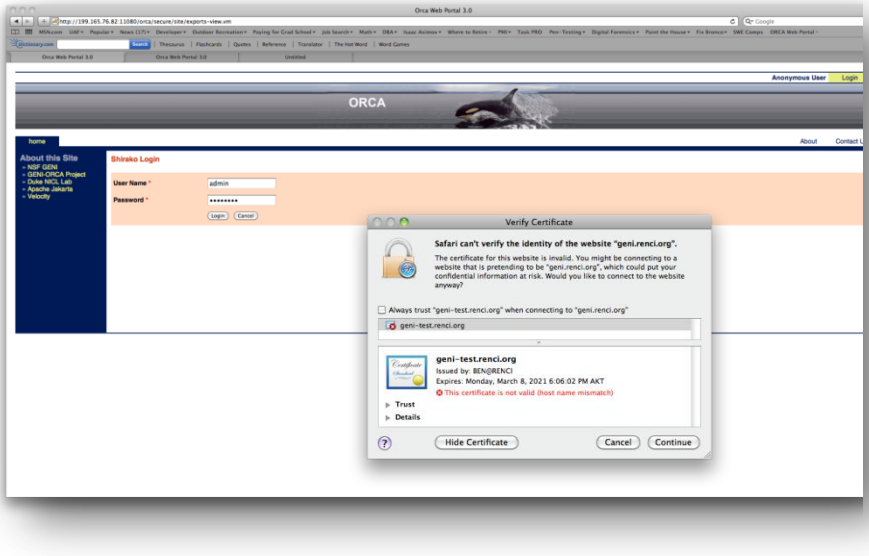


Figure 1. Logging in

Admin

In a production environment, the Principal Investigator (PI) may fill this position or delegate this responsibility to another. The administrator (admin) manages the people who fill the actor roles in ORCA, such as the uaf-vm-site, uaf-broker, and uaf-service actors. The relationships between these entities are one-to-many, with one admin to many sites, one site to many brokers, and one broker to many slice managers. In practical use, one person may fill some or all of these roles. The admin controls access by assigning new users and passwords. The admin can review users by clicking the "manage" button.

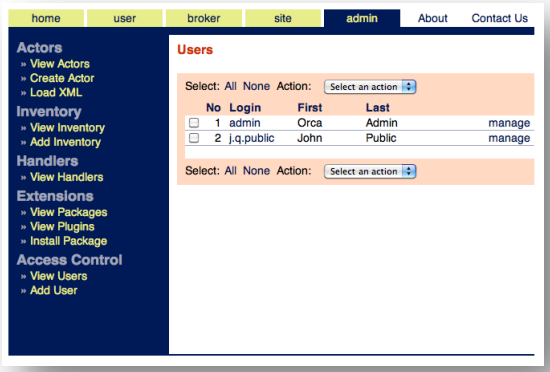


Figure 2. Added new user

Where can I get Help!

The best thing to do is join geni-orca-users@googlegroups.com now, before you need help, and send a salutation. Many people there have been a great deal of help to me, such as Ilia Baldine (the Director of Networking Research and Infrastructure, Renaissance Computing Institute), Victor J. Orlikowski, Prateek Jaipuria, and Anirban Mandal.

Figure 3. New user details

NOTE: Currently, the admin must manually set new users and passwords for the applicable container in /opt/orca/config/ container.properties. The admin must then repackage and deploy the ORCA web application (See the UAF ORCA Installation Manual). To avoid this, I will post the remaining directions as the admin only.

View Actors

The admin can view all actors in a container. Notice that the Head node lists only uaf-broker and uaf-vm-site, while the Worker node only lists uaf-service.

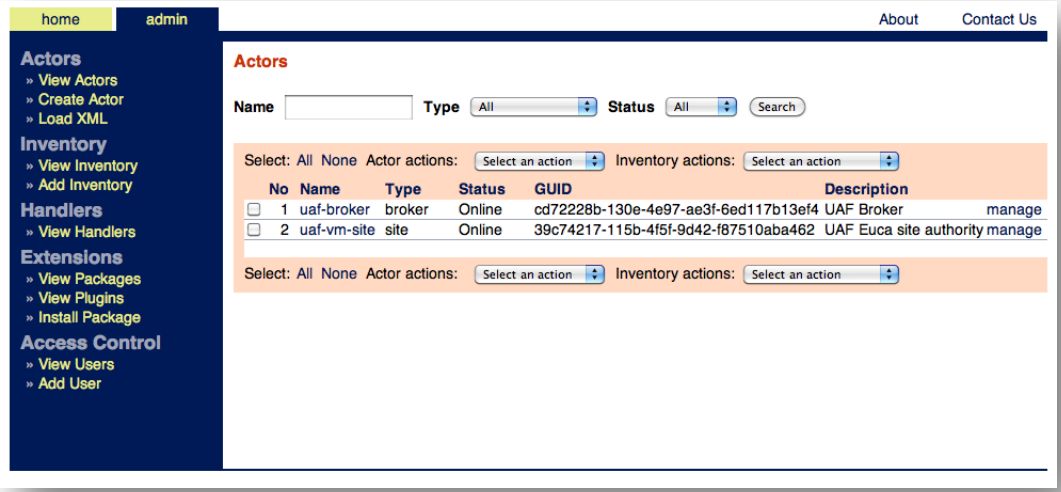


Figure 4. Actors on Head node

Site

The uaf-vm-site actor is the site authority, and it is called “vm-site” because it controls VMs. Other site authorities may exist for the same ORCA cluster, such as uaf-net-site, which we would use to allocate network resources if we were connected to National Lambda Rail (NLR), Cisco IOS Next Generation (ION) Programmable Packet Filters, or another programmable network service.

View Resource Pools

In our case, uaf-vm-site controls *UAF Euca internal vlan, UAF Gigabit Ethernet Port,* and *Eucalyptus Virtual Machine (UAF).* Only *Eucalyptus Virtual Machine (UAF)* works at this time.

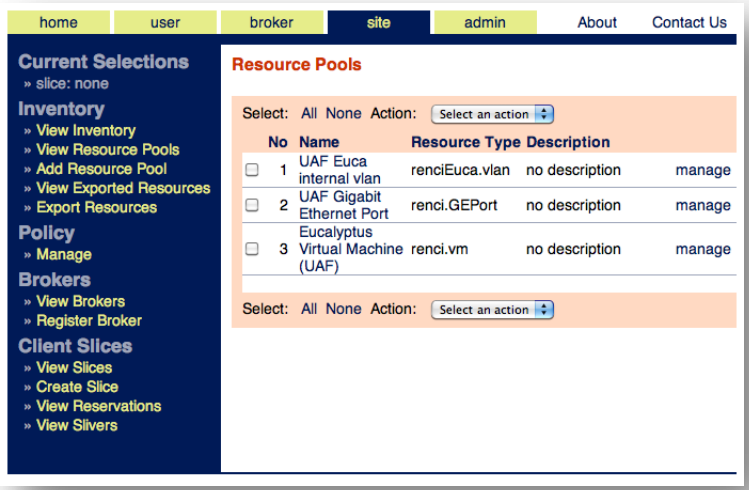


Figure 5. uaf-vm-site resource pools

Broker

A broker typically might be a PI for a project who requires ORCA resources for one or more experiments. The broker can schedule one or even all resources from the site authority for some specific time. The broker then assigns those resources to slice managers who may run the experiment or further divide the resources among users.

View Inventory

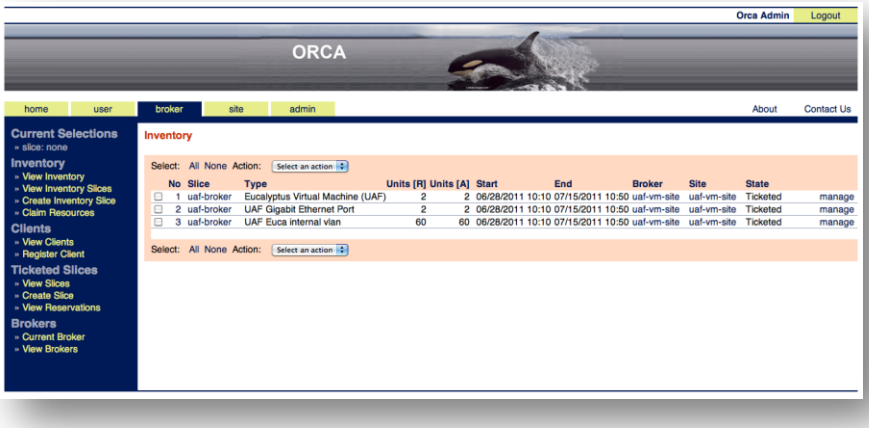


Figure 6. uaf-broker has all resources

Register Client

The broker must register at least one client (the slice manager uaf-service in this case) in order to create VMs. If uaf-service does not exist under "View Clients," then click the "Register Clients" button to add uaf-service as a client. To do so, open the url http://199.165.76.83:11080/orca in another web browser, log in, and go to the "user" tab. Under "Current Actor," you will see uaf-service actor-specific security information. Copy the *Name*, *GUID*, and *Encoded Certificate* into the appropriate blocks and click "Add."

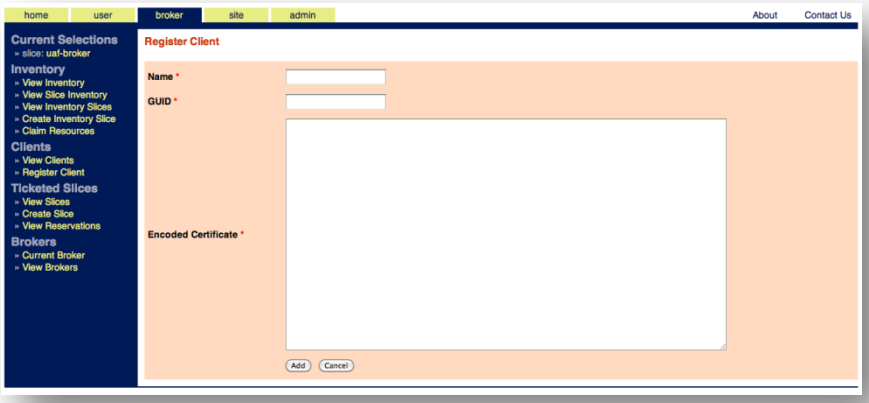


Figure 7. Register uaf-service

View Clients

Now “View Clients” should list the UAF slice manager.

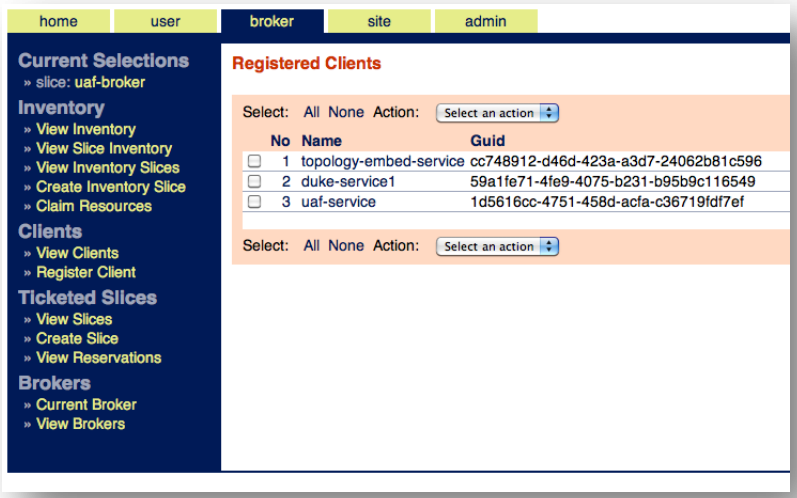


Figure 8. uaf-service as registered client

View Slices

View Slices now lists uaf-service, too.

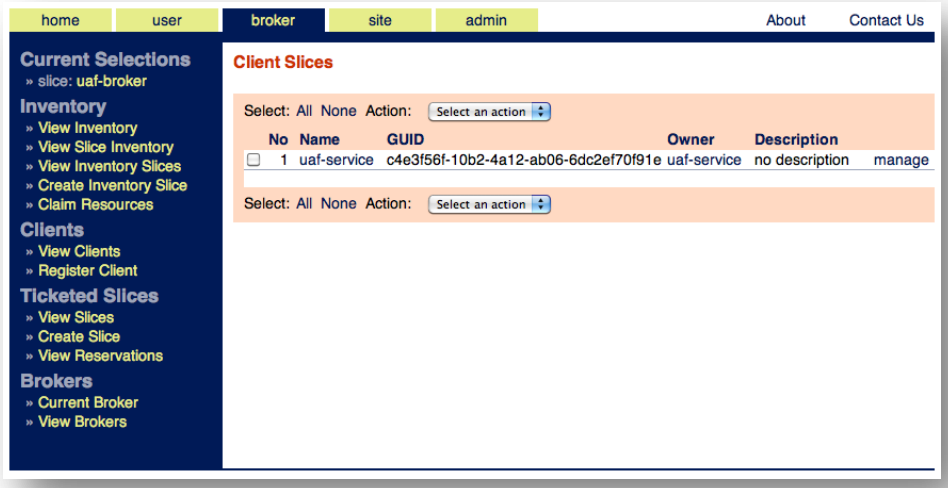


Figure 9. View Slices lists uaf-service

View Reservations

The broker also can view reservations in this tab.

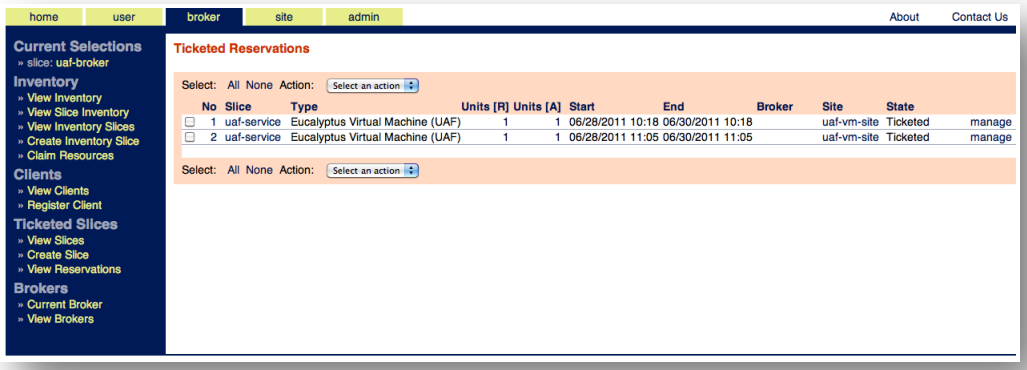


Figure 10. Reservations on uaf-broker

User

Go to http://199.165.76.83:11080/orca and open the users tab. From here, you can create reservations.

Create a UAF Reservation

Choose "Create Reservation," then "*uaf-broker*" and "*Eucalyptus Virtual Machine (UAF)*." Schedule the number of instances, the lease start, and the lease end.

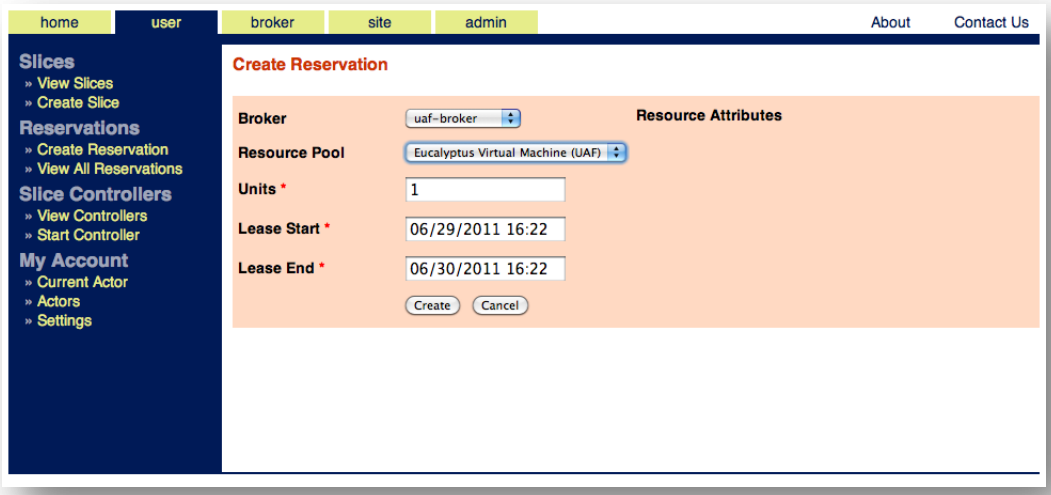


Figure 11. Create a VM on uaf-broker

View Reservations

You can now "View Reservations." Click refresh in the browser to see "Obtaining Ticket," "Redeeming Ticket," and "Active" in the "Status" column.

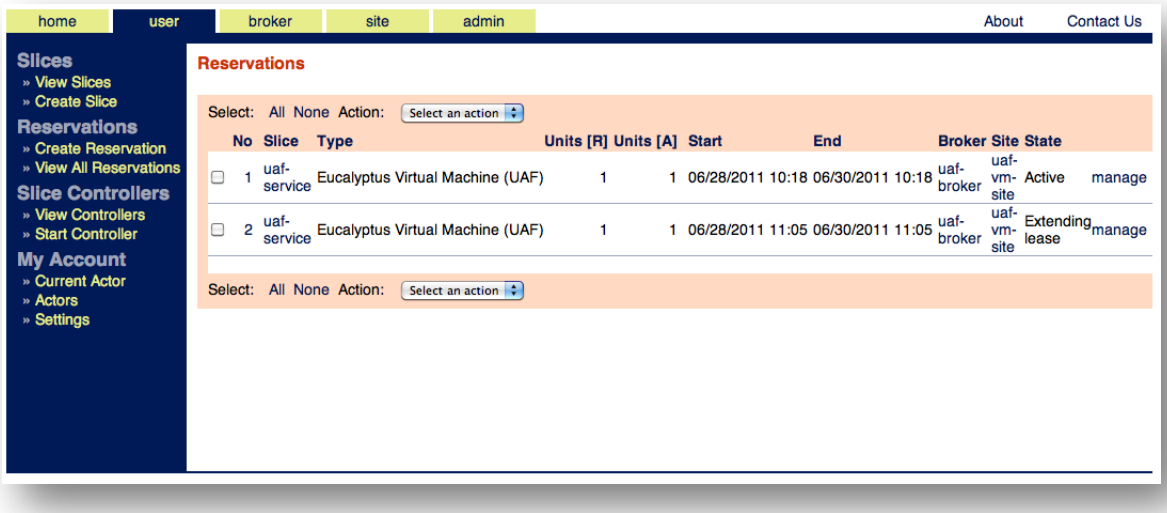


Figure 12. View all reservations

In addition, you can click manage to find out pertinent information about your instance, such as its IP address and instance ID.

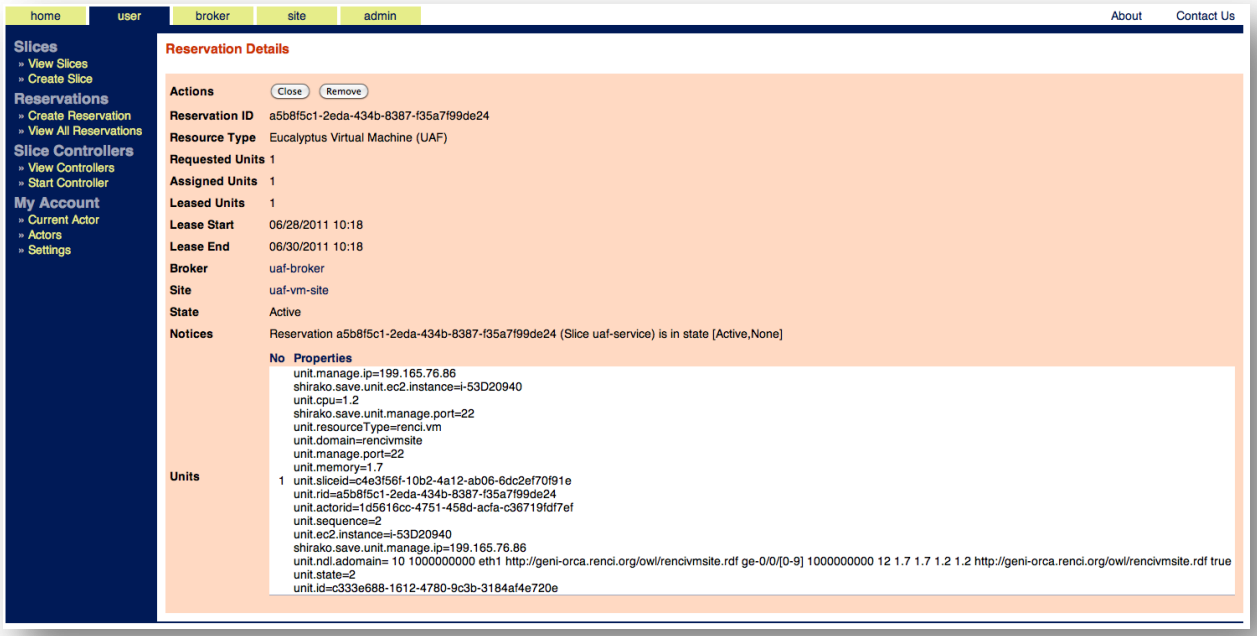


Figure 13. Reservation details

Logging into a UAF Instance

Lastly, log into the instance by using a this command in a terminal: `ssh –i mykey.private root@199.165.76.86`

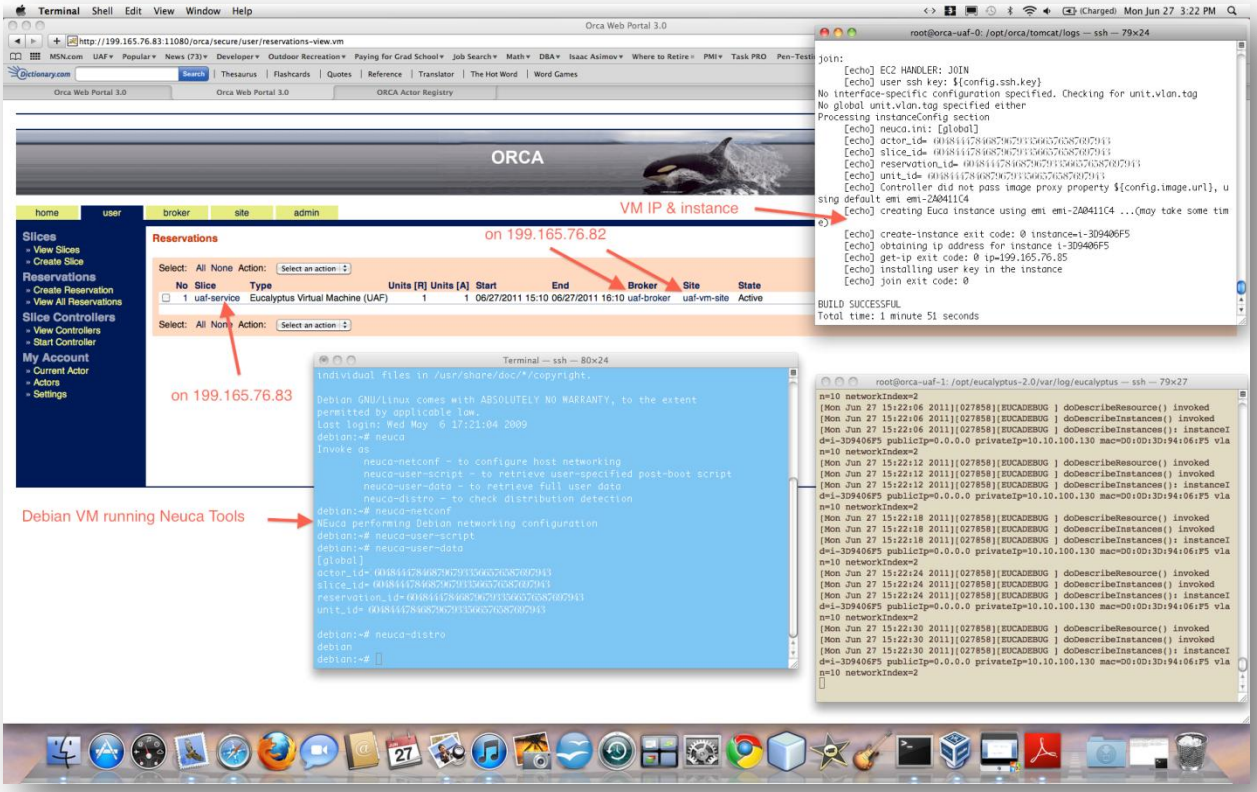


Figure 14. Logging into instance

Connecting to Other ORCA clusters

With ORCA, one can request resources from other ORCA cluster at Duke University, UNC, RENCI, and other institutions. Use these directions https://geni-orca.renci.org/trac/wiki/orca-xmlrpc-controller to create virtual machines in other ORCA clusters.

XML-RPC

Log into uaf-service User tab and select "Start Controller," then select "XML-RPC controller" and "Create." You now have a running XML-RPC controller.

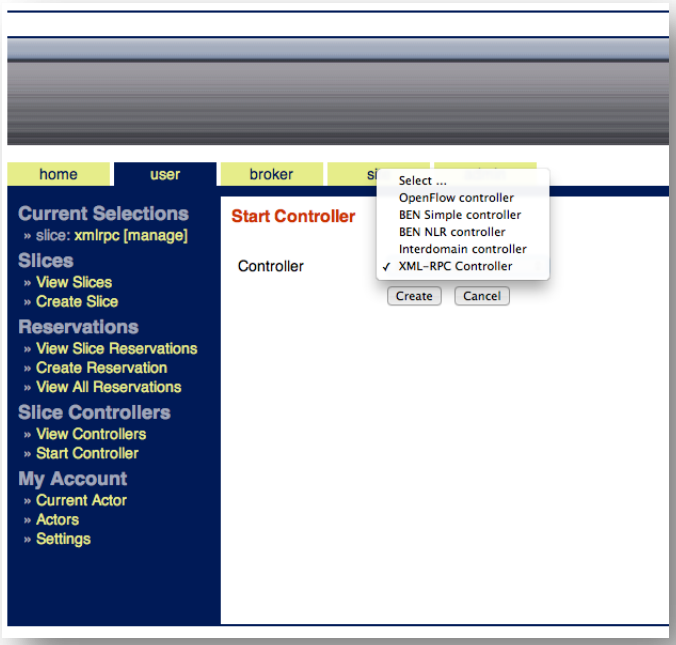


Figure 15. Starting an XML-RPC controller

Now, select "View Controllers"

Figure 16. View XML-RPC controller

Now selct “View Slices” to view your XML-RPC slice.

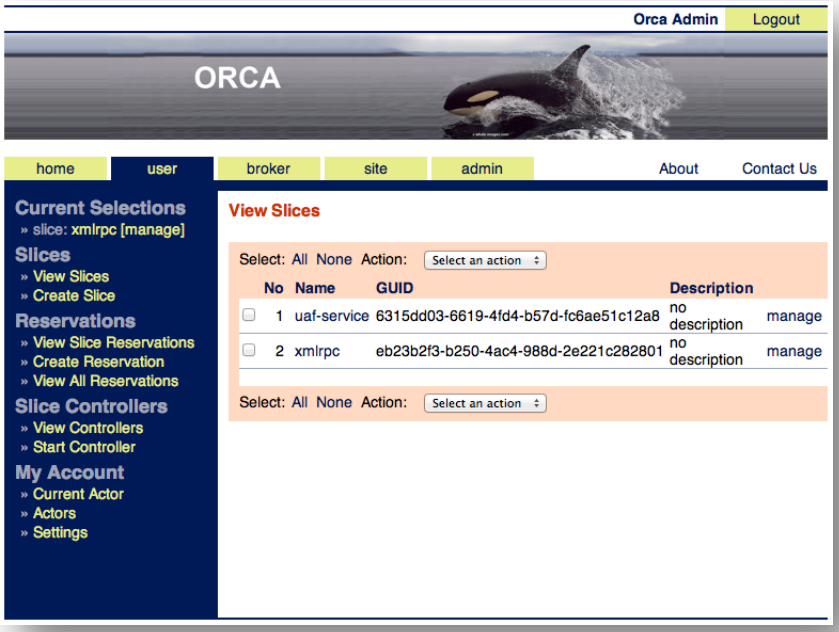


Figure 17. View XML-RPC slice

Using GENI AM API controller plugin

Now you can use the python scripts to populate your XML-RPC slice with components. The scripts are located in `$ORCA_HOME/controllers/xmlrpc/resources/scripts`. For instance,

```
root@orca-uaf-0:/home/orca/orca/controllers/xmlrpc/resources/scripts# python
GetVersion.py -s http://199.165.76.84:11080/orca/xmlrpc

Querying ORCA xml-rpc server for current AM API version ...

Current API version = {'implementation': 'ORCA', 'geni_api': 1}
```

Creating a XML-RPC Reservaton

This section describes a simple means of creating and populating a XML-RPC slice. ORCA will decide what clusters to embed into based on resource availability. Use “View Slice Reservations” or “View All Reservations” to get the sliver status. Clicking “Manage” reports the IP addresses and port numbers of the management interfaces.

These steps explain how to use uaf-service to request resources from some ORCA cluster. The RSpec below, given to me by Ilia Baldine, requests two instances connected by one internal VLan.

RSpec to NDL conversion

The easiest way to get an NDL request file is to start with RSpec for now. Take something like this (a request for two nodes with a link between them):

Table 2. XML-RPC RSpec

```
<?xml version="1.0" encoding="UTF-8"?>
<rspec type="request"
xsi:schemaLocation="http://www.protogeni.net/resources/rspec/2

http://www.protogeni.net/resources/rspec/2/request.xsd"
      xmlns:flack="http://www.protogeni.net/resources/rspec/ext/flack/1"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns="http://www.protogeni.net/resources/rspec/2">
 <node client_id="geni1">
   <sliver_type name="raw-pc">
     <disk_image
name="http://geni-images.renci.org/images/gush/gush-deb5-i386.xml"
version="25f53b64cfe44dd1604447f04b7b533bb67dea1e" />
   </sliver_type>
   <services>
     <execute shell="sh" command="sudo hostname `cat
/var/emulab/boot/realname`.`cat /var/emulab/boot/mydomain`"/>
   </services>
   <interface client_id="geni1:0">
      <ip address="172.16.1.1" netmask="255.255.0.0" />
   </interface>
 </node>
 <node client_id="geni2">
   <sliver_type name="raw-pc">
     <disk_image
name="http://geni-images.renci.org/images/gush/gush-deb5-i386.xml"
version="25f53b64cfe44dd1604447f04b7b533bb67dea1e" />
   </sliver_type>
   <services>
     <execute shell="sh" command="sudo hostname `cat
/var/emulab/boot/realname`.`cat /var/emulab/boot/mydomain`"/>
   </services>
   <interface client_id="geni2:0" >
     <ip address="172.16.1.2" netmask="255.255.0.0" />
   </interface>
 </node>
  <link client_id="center">
    <interface_ref client_id="geni1:0" />
    <interface_ref client_id="geni2:0" />
  </link>
</rspec>
```

Then run it through the converter (select RSpec v2 request and RDF-XML as output) in your browser: http://geni-test.renci.org:11080/ndl-conversion/convert.jsp

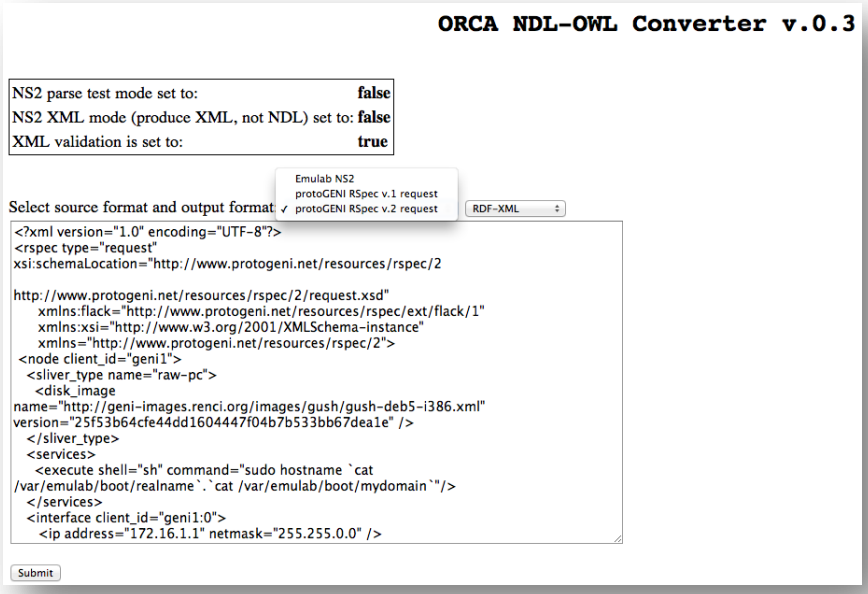


Figure 18. ORCA NDL-OWL Converter

This converts the RSpec to the NDL file below:

Table 3. NDL file converted from RSpec

```
<rdf:RDF
    xmlns:compute="http://geni-orca.renci.org/owl/compute.owl#"
    xmlns:request="http://geni-orca.renci.org/owl/8e772971-0868-4cb0-91d9-
68b651bc5d23#"
    xmlns:ethernet="http://geni-orca.renci.org/owl/ethernet.owl#"
    xmlns:time="http://www.w3.org/2006/time#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:layer="http://geni-orca.renci.org/owl/layer.owl#"
    xmlns:ip4="http://geni-orca.renci.org/owl/ip4.owl#"
    xmlns:orca="http://geni-orca.renci.org/owl/orca.owl#"
    xmlns:request-schema="http://geni-orca.renci.org/owl/request.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:collections="http://geni-orca.renci.org/owl/collections.owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:topology="http://geni-orca.renci.org/owl/topology.owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" >
  <rdf:Description rdf:about="http://geni-orca.renci.org/owl/8e772971-0868-4cb0-91d9-
68b651bc5d23#Term">
    <time:hasDurationDescription rdf:resource="http://geni-
orca.renci.org/owl/8e772971-0868-4cb0-91d9-68b651bc5d23#TermDuration"/>
    <rdf:type rdf:resource="http://www.w3.org/2006/time#Interval"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://geni-orca.renci.org/owl/8e772971-0868-4cb0-91d9-
68b651bc5d23#ip-172-16-1-2">
    <ip4:netmask>255.255.0.0</ip4:netmask>
    <layer:label_ID>172.16.1.2</layer:label_ID>
    <rdf:type rdf:resource="http://geni-orca.renci.org/owl/ip4.owl#IPAddress"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://geni-orca.renci.org/owl/8e772971-0868-4cb0-91d9-
68b651bc5d23#ip-172-16-1-1">
```

```
    <ip4:netmask>255.255.0.0</ip4:netmask>
    <layer:label_ID>172.16.1.1</layer:label_ID>
    <rdf:type rdf:resource="http://geni-orca.renci.org/owl/ip4.owl#IPAddress"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://geni-orca.renci.org/owl/8e772971-0868-4cb0-91d9-
68b651bc5d23#center">
    <topology:hasInterface rdf:resource="http://geni-orca.renci.org/owl/8e772971-0868-
4cb0-91d9-68b651bc5d23#geni2-0"/>
    <topology:hasInterface rdf:resource="http://geni-orca.renci.org/owl/8e772971-0868-
4cb0-91d9-68b651bc5d23#geni1-0"/>
    <rdf:type rdf:resource="http://geni-
orca.renci.org/owl/topology.owl#NetworkConnection"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://geni-orca.renci.org/owl/8e772971-0868-4cb0-91d9-
68b651bc5d23#geni1-0">
    <ip4:localIPAddress rdf:resource="http://geni-orca.renci.org/owl/8e772971-0868-
4cb0-91d9-68b651bc5d23#ip-172-16-1-1"/>
    <topology:hostInterfaceName>0</topology:hostInterfaceName>
    <rdf:type rdf:resource="http://geni-orca.renci.org/owl/topology.owl#Interface"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://geni-orca.renci.org/owl/8e772971-0868-4cb0-91d9-
68b651bc5d23#25f53b64cfe44dd1604447f04b7b533bb67dea1e">
    <topology:hasURL>http://geni-images.renci.org/images/gush/gush-deb5-
i386.xml</topology:hasURL>
    <topology:hasGUID>25f53b64cfe44dd1604447f04b7b533bb67dea1e</topology:hasGUID>
    <rdf:type rdf:resource="http://geni-orca.renci.org/owl/compute.owl#VMImage"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://geni-orca.renci.org/owl/8e772971-0868-4cb0-91d9-
68b651bc5d23#geni2-0">
    <ip4:localIPAddress rdf:resource="http://geni-orca.renci.org/owl/8e772971-0868-
4cb0-91d9-68b651bc5d23#ip-172-16-1-2"/>
    <topology:hostInterfaceName>0</topology:hostInterfaceName>
    <rdf:type rdf:resource="http://geni-orca.renci.org/owl/topology.owl#Interface"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://geni-orca.renci.org/owl/8e772971-0868-4cb0-91d9-
68b651bc5d23#TermDuration">
    <time:hours
rdf:datatype="http://www.w3.org/2001/XMLSchema#decimal">24</time:hours>
    <rdf:type rdf:resource="http://www.w3.org/2006/time#DurationDescription"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://geni-orca.renci.org/owl/8e772971-0868-4cb0-91d9-
68b651bc5d23#geni1">
    <topology:hasInterface rdf:resource="http://geni-orca.renci.org/owl/8e772971-0868-
4cb0-91d9-68b651bc5d23#geni1-0"/>
    <compute:hasVMImage rdf:resource="http://geni-orca.renci.org/owl/8e772971-0868-
4cb0-91d9-68b651bc5d23#25f53b64cfe44dd1604447f04b7b533bb67dea1e"/>
    <rdf:type rdf:resource="http://geni-orca.renci.org/owl/compute.owl#Server"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://geni-orca.renci.org/owl/8e772971-0868-4cb0-91d9-
68b651bc5d23#">
    <collections:element rdf:resource="http://geni-orca.renci.org/owl/8e772971-0868-
4cb0-91d9-68b651bc5d23#center"/>
    <collections:element rdf:resource="http://geni-orca.renci.org/owl/8e772971-0868-
4cb0-91d9-68b651bc5d23#geni2"/>
    <compute:hasVMImage rdf:resource="http://geni-orca.renci.org/owl/8e772971-0868-
4cb0-91d9-68b651bc5d23#25f53b64cfe44dd1604447f04b7b533bb67dea1e"/>
    <collections:element rdf:resource="http://geni-orca.renci.org/owl/8e772971-0868-
4cb0-91d9-68b651bc5d23#geni1"/>
    <request-schema:hasTerm rdf:resource="http://geni-orca.renci.org/owl/8e772971-
0868-4cb0-91d9-68b651bc5d23#Term"/>
    <rdf:type rdf:resource="http://geni-orca.renci.org/owl/request.owl#Reservation"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://geni-orca.renci.org/owl/8e772971-0868-4cb0-91d9-
```

```
68b651bc5d23#geni2">
    <topology:hasInterface rdf:resource="http://geni-orca.renci.org/owl/8e772971-0868-
4cb0-91d9-68b651bc5d23#geni2-0"/>
    <compute:hasVMImage rdf:resource="http://geni-orca.renci.org/owl/8e772971-0868-
4cb0-91d9-68b651bc5d23#25f53b64cfe44dd1604447f04b7b533bb67dea1e"/>
    <rdf:type rdf:resource="http://geni-orca.renci.org/owl/compute.owl#Server"/>
  </rdf:Description>
</rdf:RDF>
```

Save the output as `PracticeNDLNodeLink.xml`, for example, and use it in createSliver.py. Notice that you use a public key to create a new sliver, and not the eucalyptus private key `mykey.private`.

```
root@orca-uaf-0:/home/orca/orca/controllers/xmlrpc/resources/scripts# python
createSliver.py -s http://199.165.76.84:11080/orca/xmlrpc -k ~/.ssh/id_rsa.pub
-i xmlrpc -r /home/orca/PracticeNDLNodeLink.xml

[RDF]

      …

[ssh key]

      …

Contacting ORCA xml-rpc server http://199.165.76.84:11080/orca/xmlrpc for
creating the sliver...

Waiting for sliver details...

Request id: df7498d7-7e3a-4758-bd01-7bba50078440

[   Slice UID: e3225d58-c588-4421-ae7d-56701ae46507 | Reservation UID:
e1caff5b-dd66-499a-915c-2bf757936526 | Resource Type: dukeEuca.vlan | Resource
Units: 1 ]

[   Slice UID: e3225d58-c588-4421-ae7d-56701ae46507 | Reservation UID:
dc998624-5ae9-42c8-9189-1e162df35221 | Resource Type: duke.vm | Resource Units:
1 ]

[   Slice UID: e3225d58-c588-4421-ae7d-56701ae46507 | Reservation UID:
4d8494e4-e111-4df1-9f30-fce2a1bb72fc | Resource Type: duke.vm | Resource Units:
1 ]

Use Slice UID to check status of the sliver, renew sliver lease or delete the
sliver
```

XML-RPC Sliver Status and Removal

To check the status of the sliver, run the 'sliverStatus' script

```
root@orca-uaf-0:/home/orca/orca/controllers/xmlrpc/resources/scripts# python
sliverStatus.py -s http://199.165.76.84:11080/orca/xmlrpc -i xmlrpc
```

To delete the sliver, run the 'deleteSliver' script.

```
root@orca-uaf-0:/home/orca/orca/controllers/xmlrpc/resources/scripts# python
deleteSliver.py -s http://199.165.76.84:11080/orca/xmlrpc -i xmlrpc
```

Logging Into a XML-RPC Slice Instance

Use the private key that you used to create the xmlrpc sliver to log into the instance by gathering the unit.manage.port and unit.manage.ip from the “manage” button after the reservation is “Active.”

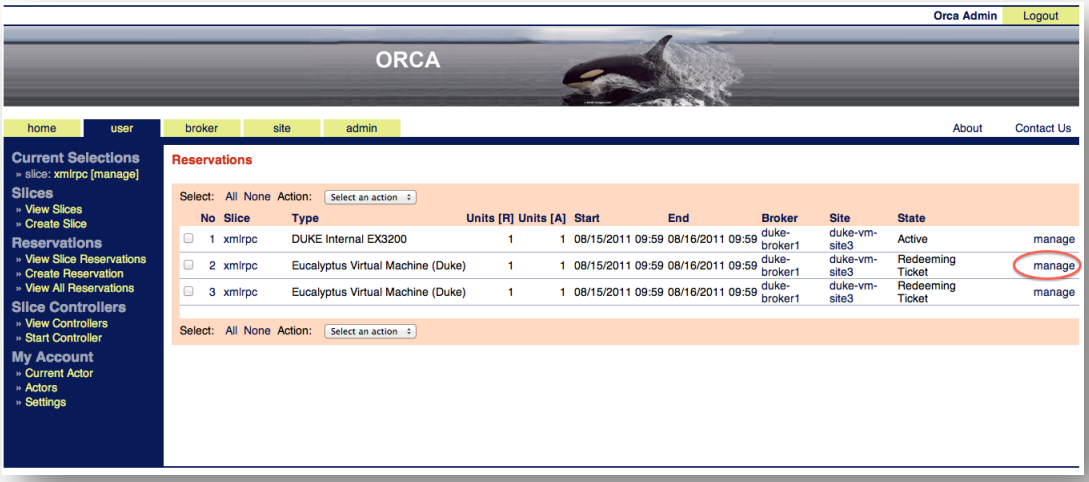


Figure 19. Manage XML-RPC instance

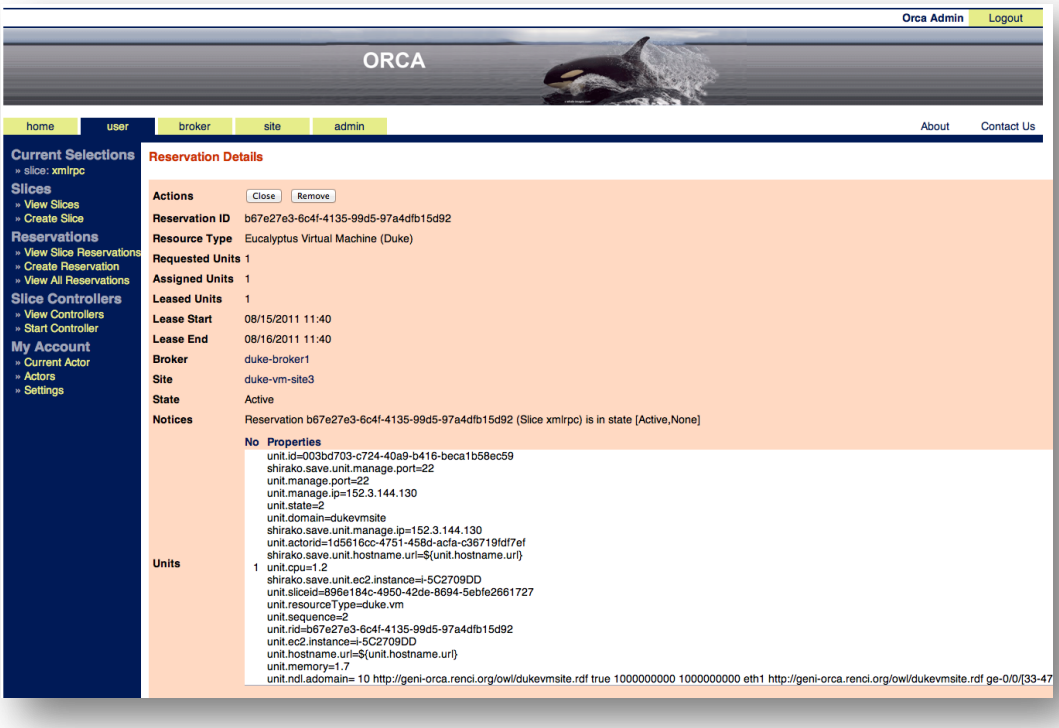


Figure 20. XML-RPC instance properties

Log into the instances you created with the matching private key to `id_rsa.pub` in `createSliver.py`.

```
root@orca-uaf-0:/home/orca/orca/controllers/xmlrpc/resources/scripts# ssh -i
/root/.ssh/id_rsa -p 22 root@152.3.144.131
```

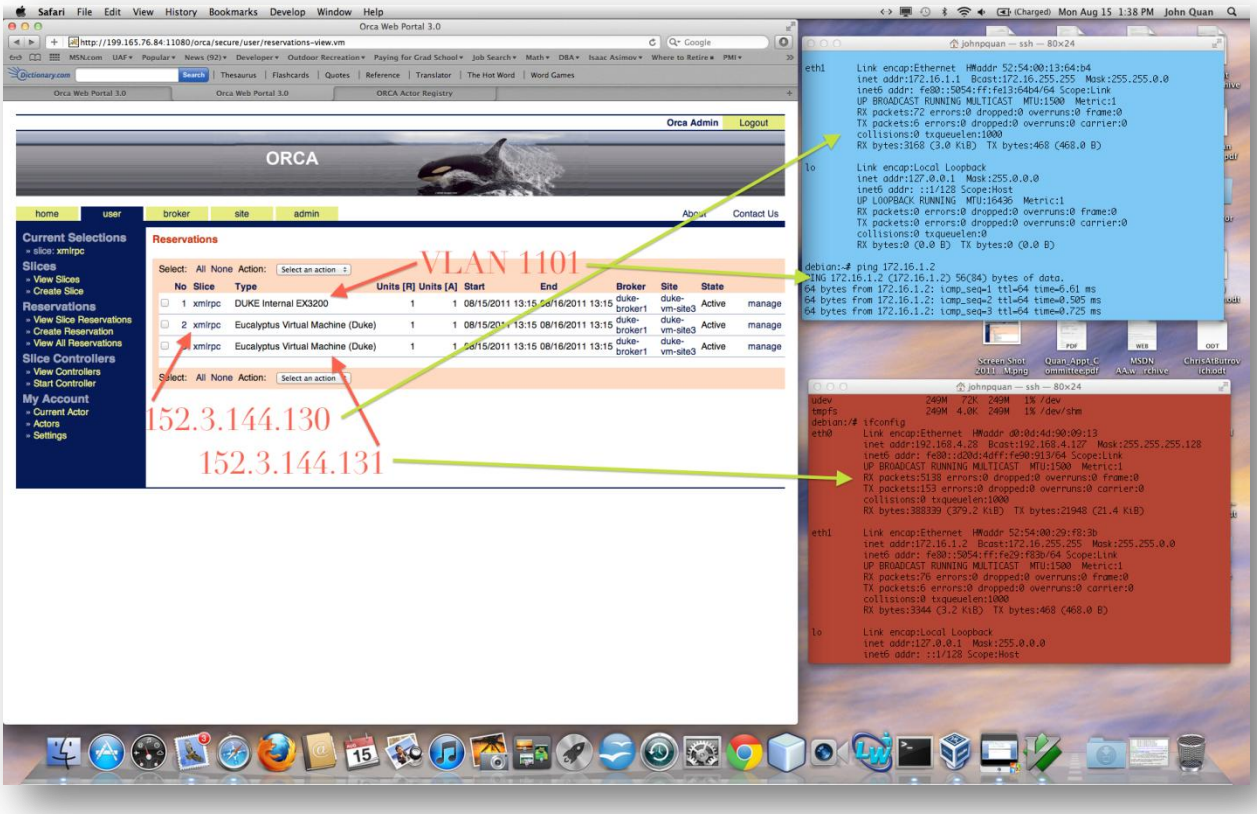


Figure 21. Simple XML-RPC slice