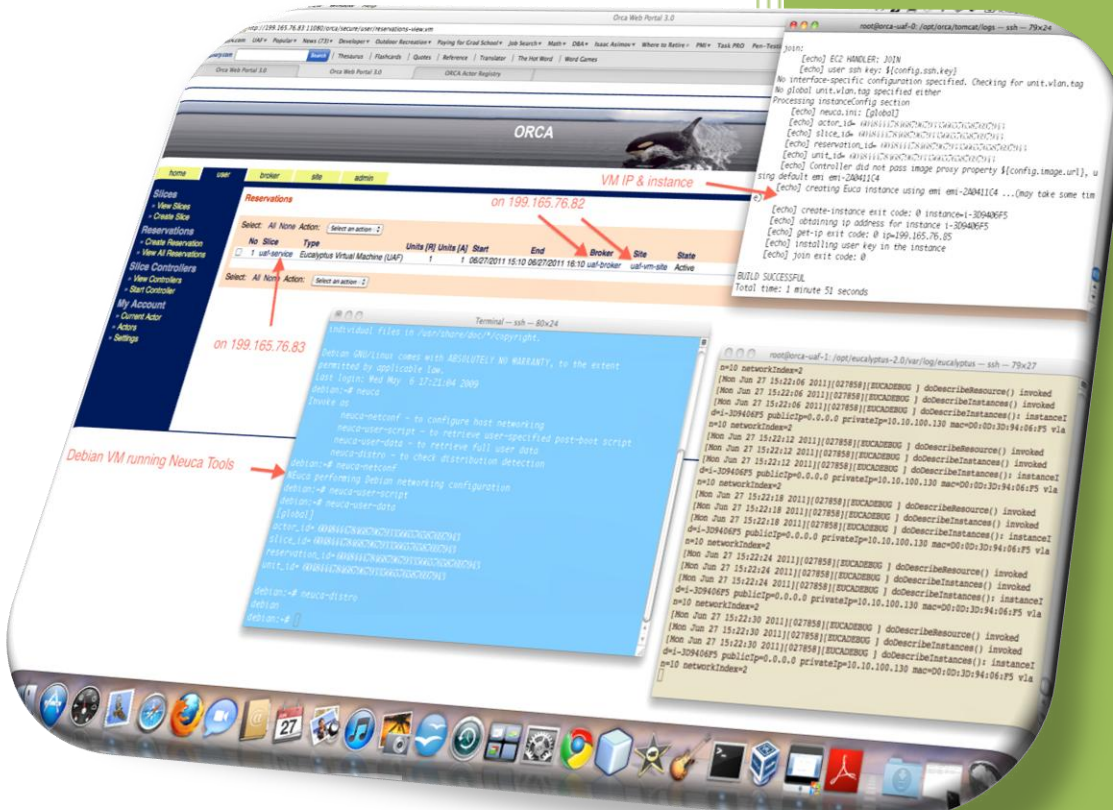


GENI

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	4
Purpose	4
System Setup.....	5
Image Proxy.....	5
Head and Worker Nodes.....	5
The Actor Registry	7
Log In	7
Admin	8
Where can I get Help!.....	8
View Actors	9
Site.....	10
View Resource Pools.....	10
Broker	11
View Inventory	11
Register Client	11
View Clients.....	12
View Slices.....	12
View Reservations	12
User	13
Create a UAF Reservation	13
View Reservations	13
Logging into a UAF Instance.....	15
Connecting to Other ORCA clusters	16
XML-RPC.....	16
Using GENI AM API controller plugin.....	17
Creating a XML-RPC Reservaton.....	17
RSpec to NDL conversion	17
XML-RPC Sliver Status and Removal.....	21
Logging Into a XML-RPC Slice Instance	22
Bibliography	Error! Bookmark not defined.

Table 1. Basic configuration	5
Table 2. XML-RPC RSpec	18
Table 3. NDL file converted from RSpec	19
Figure 1. Logging in.....	7
Figure 2. Added new user	8
Figure 3. New user details	8
Figure 4. Actors on Head node.....	9
Figure 5. uaf-vm-site resource pools	10
Figure 6. uaf-broker has all resources	11
Figure 7. Register uaf-service.....	11
Figure 8. uaf-service as registered client.....	12
Figure 9. View Slices lists uaf-service	12
Figure 10. Reservations on uaf-broker	12
Figure 11. Create a VM on uaf-broker	13
Figure 12. View all reservations	13
Figure 13. Reservation details.....	14
Figure 14. Logging into instance.....	15
Figure 15. Starting an XML-RPC controller	16
Figure 16. View XML-RPC controller.....	16
Figure 17. View XML-RPC slice	17
Figure 18. ORCA NDL-OWL Converter	19
Figure 19. Manage XML-RPC instance	22
Figure 20. XML-RPC instance properties	22
Figure 21. Simple XML-RPC slice	23

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 Ubuntu¹ 11.04 Natty Narwhal operating system, with Image Proxy² installed on the Head Node, and ORCA³ Camano 3.0 and Eucalyptus⁴ 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.

							tsPolicy	Key				
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-4751-7458d-acfac36719fdf7ef	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.AuthorityCalendarPolicy	orca.policy.core.AuthorityCalendarPolicy	Click for Public Key	Click for Actor Certificate	Click for Abstract Site NDL	Click for Full Site NDL	

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.

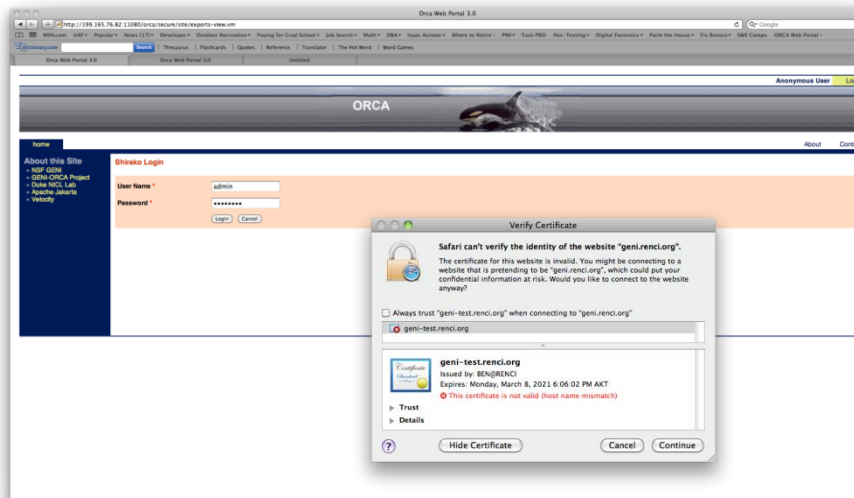


Figure 1. Logging in

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.

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.

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.

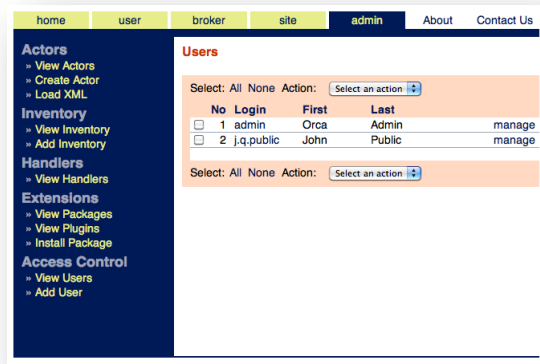


Figure 2. Added new user

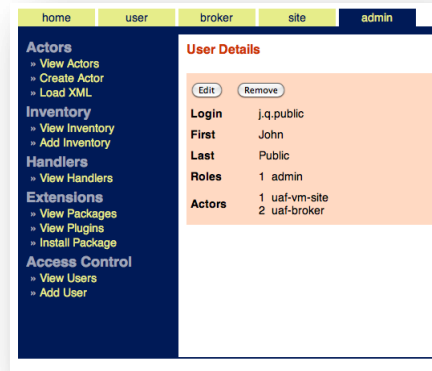


Figure 3. New user details

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.

The screenshot displays the 'Actors' management page. The left sidebar contains navigation links for 'Actors', 'Inventory', 'Handlers', 'Extensions', and 'Access Control'. The main content area shows a table of actors with the following data:

No	Name	Type	Status	GUID	Description	
<input type="checkbox"/>	1	uaf-broker	broker	Online	cd72228b-130e-4e97-ae3f-6ed117b13ef4	UAF Broker manage
<input type="checkbox"/>	2	uaf-vm-site	site	Online	39c74217-115b-4f5f-9d42-f87510aba462	UAF Euca site authority manage

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.

The screenshot shows a web interface for managing resource pools. The top navigation bar includes links for home, user, broker, site, admin, About, and Contact Us. The left sidebar contains a navigation menu with sections: Current Selections (slice: none), Inventory (View Inventory, View Resource Pools, Add Resource Pool, View Exported Resources, Export Resources), Policy (Manage), Brokers (View Brokers, Register Broker), and Client Slices (View Slices, Create Slice, View Reservations, View Silvers). The main content area is titled "Resource Pools" and features a table with the following data:

No	Name	Resource Type	Description	
<input type="checkbox"/>	1 UAF Euca internal vlan	renciEuca.vlan	no description	manage
<input type="checkbox"/>	2 UAF Gigabit Ethernet Port	renci.GEPort	no description	manage
<input type="checkbox"/>	3 Virtual Machine (UAF)	renci.vm	no description	manage

Below the table, there are two "Select: All None Action:" labels, each followed by a "Select an action" dropdown menu.

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

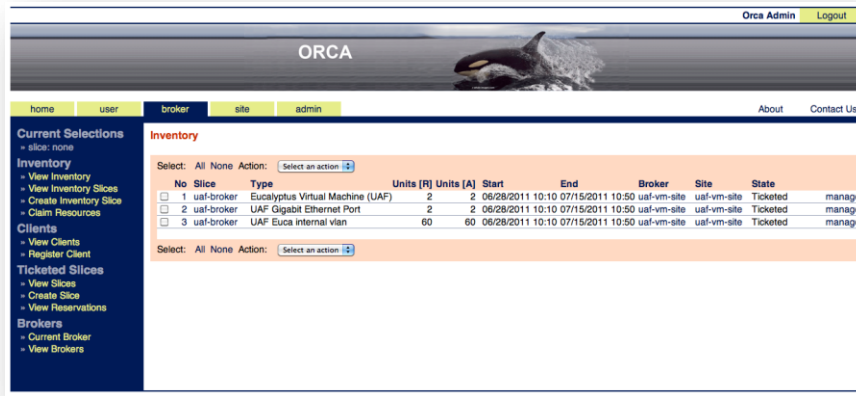


Figure 6. uaf-broker has all resources

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.”

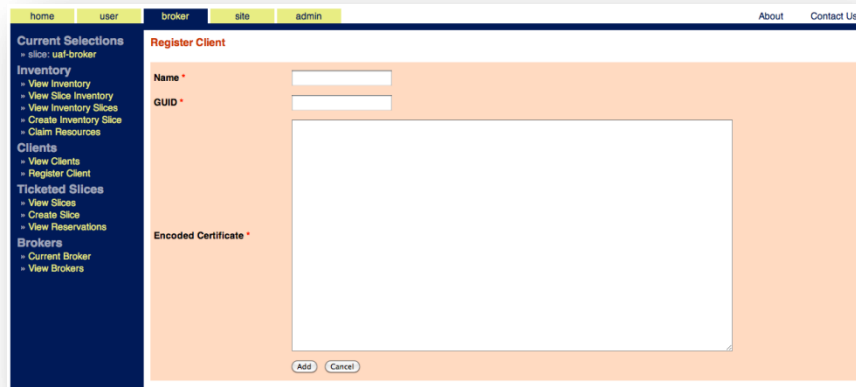


Figure 7. Register uaf-service

View Clients

Now "View Clients" should list the UAF slice manager.

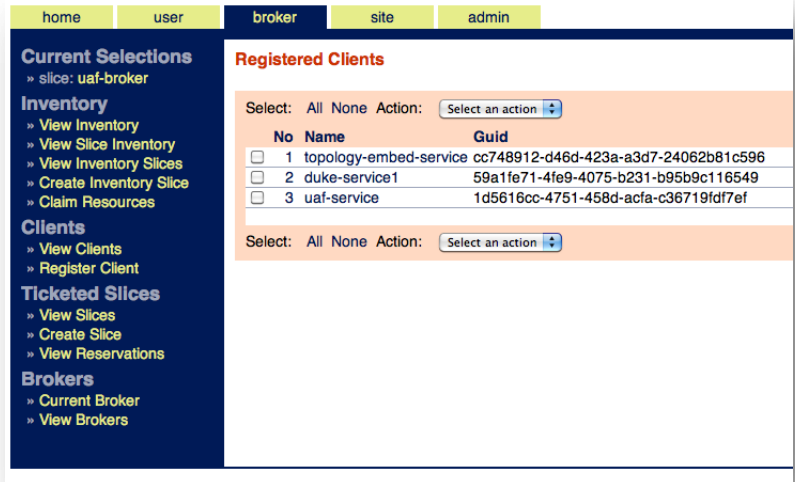


Figure 8. uaf-service as registered client

View Slices

View Slices now lists uaf-service, too.

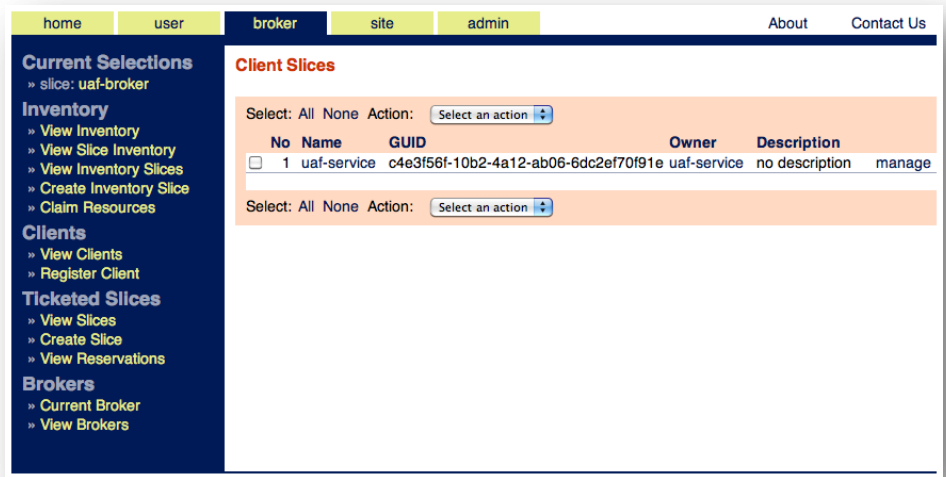


Figure 9. View Slices lists uaf-service

View Reservations

The broker also can view reservations in this tab.

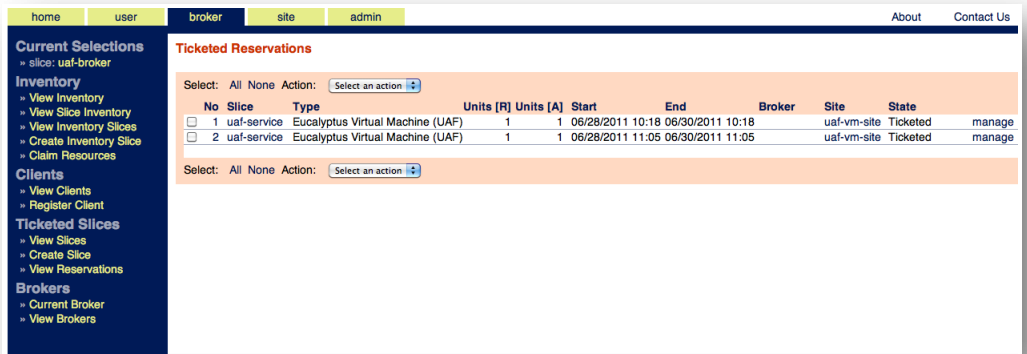


Figure 10. Reservations on uaf-broker

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.

The screenshot shows the 'Create Reservation' form in the ORCA web interface. The form is titled 'Create Reservation' and is located in the 'user' tab. The form has a light orange background and contains the following fields:

- Broker:** A dropdown menu with 'uaf-broker' selected.
- Resource Pool:** A dropdown menu with 'Eucalyptus Virtual Machine (UAF)' selected.
- Units *:** A text input field with '1' entered.
- Lease Start *:** A text input field with '06/29/2011 16:22' entered.
- Lease End *:** A text input field with '06/30/2011 16:22' entered.

At the bottom of the form, there are two buttons: 'Create' and 'Cancel'. The left sidebar contains navigation links for Slices, Reservations, Slice Controllers, and My Account.

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.

The screenshot shows the 'Reservations' table in the ORCA web interface. The table is titled 'Reservations' and is located in the 'user' tab. The table has a light orange background and contains the following columns:

- No:** Reservation number.
- Slice:** Slice name.
- Type:** Resource type.
- Units [R]:** Requested units.
- Units [A]:** Allocated units.
- Start:** Start time.
- End:** End time.
- Broker:** Broker name.
- Site:** Site name.
- State:** Reservation state.
- manage:** Action link.

The table contains two rows of reservations:

No	Slice	Type	Units [R]	Units [A]	Start	End	Broker	Site	State	manage
1	uaf-service	Eucalyptus Virtual Machine (UAF)	1	1	06/28/2011 10:18	06/30/2011 10:18	uaf-broker	uaf-vm-site	Active	manage
2	uaf-service	Eucalyptus Virtual Machine (UAF)	1	1	06/28/2011 11:05	06/30/2011 11:05	uaf-broker	uaf-vm-site	Extending lease	manage

At the top of the table, there is a 'Select:' dropdown menu with 'All' selected and an 'Action:' dropdown menu with 'Select an action' selected. At the bottom of the table, there is another 'Select:' dropdown menu with 'All' selected and an 'Action:' dropdown menu with 'Select an action' selected.

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.

The screenshot displays a web interface with a navigation menu on the left and a main content area. The navigation menu includes sections for Slices, Reservations, Slice Controllers, and My Account. The main content area is titled "Reservation Details" and shows the following information:

- Actions:** Close, Remove
- Reservation ID:** a5b8f5c1-2eda-434b-8387-f35a7f99de24
- Resource Type:** Eucalyptus Virtual Machine (UAF)
- Requested Units:** 1
- Assigned Units:** 1
- Leased Units:** 1
- Lease Start:** 06/28/2011 10:18
- Lease End:** 06/30/2011 10:18
- Broker:** uaf-broker
- Site:** uaf-vm-site
- State:** Active
- Notices:** Reservation a5b8f5c1-2eda-434b-8387-f35a7f99de24 (Slice uaf-service) is in state [Active,None]
- Properties:**
 - unit.manage.ip=199.165.76.86
 - shirako.save.unit.ec2.instance=i-53D20940
 - unit.cpu=1.2
 - shirako.save.unit.manage.port=22
 - unit.resourceType=renci-vm
 - unit.domain=rencivmsite
 - unit.manage.port=22
 - unit.memory=1.7
- Units:** 1
 - unit.sliceid=c4e3f56f-10b2-4a12-ab06-6dc2ef70f91e
 - unit.rid=a5b8f5c1-2eda-434b-8387-f35a7f99de24
 - unit.actorid=1d5616cc-4751-458d-acta-c36719fd7ef
 - unit.sequence=2
 - unit.ec2.instance=i-53D20940
 - shirako.save.unit.manage.ip=199.165.76.86
 - unit.ndi.adomain= 10 1000000000 eth1 http://geni-orca.renci.org/owl/rencivmsite.rdf ge-0/0/[0-9] 1000000000 12 1.7 1.7 1.2 1.2 http://geni-orca.renci.org/owl/rencivmsite.rdf true
 - unit.state=2
 - unit.id=c333e688-1612-4780-9c3b-3184af4e720e

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`

The screenshot displays the ORCA web portal interface with several terminal windows overlaid. The web portal shows a table of reservations with the following data:

No	Slice	Type	Units [R] Units [A]	Start	End	Broker	Site	State
1	uaaf-service	Eucalyptus Virtual Machine (UAF)	1	08/27/2011 15:10	08/27/2011 16:10	uaaf-broker	uaaf-vm-site	Active

Annotations on the screenshot include:

- A red arrow pointing to the 'uaaf-service' slice in the table, labeled "VM IP & instance on 199.165.76.82".
- A red arrow pointing to the 'uaaf-broker' column, labeled "on 199.165.76.83".
- A red arrow pointing to the terminal window showing the execution of `neuca-netconf`, labeled "Debian VM running Neuca Tools".

The terminal windows show the following content:

```
root@orca-uaaf-0: /opt/orca/tomcat/logs - ssh - 79x24
join:
[echo] EC2_HANDLER: JOIN
[echo] user ssh key: ${config.ssh.key}
No interface-specific configuration specified. Checking for unit.vlan.tag
No global unit.vlan.tag specified either
Processing InstanceConfig section
[echo] neuca.ini: [global]
[echo] actor_id=0488447846879079133663703070943
[echo] slice_id=0488447846879079133663703070943
[echo] reservation_id=0488447846879079133663703070943
[echo] unit_id=0488447846879079133663703070943
[echo] Controller did not pass image proxy property ${config.image.url}, u
sing default emt-2404114
[echo] creating Eucal instance using emt-2404114 ... (may take some tin
[echo] create-instance exit code: 0 instance=i-309406f5
[echo] obtaining ip address for instance i-309406f5
[echo] get-ip exit code: 0 ip=199.165.76.85
[echo] installing user key in the instance
[echo] join exit code: 0

BUILD SUCCESSFUL
Total time: 1 minute 51 seconds

root@orca-uaaf-1: /opt/eucalyptus-2.0/var/log/eucalyptus - ssh - 79x27
n=10 networkIndex=2
[Mon Jun 27 15:22:06 2011][027858][EUCADBEG] doDescribeResource() invoked
[Mon Jun 27 15:22:06 2011][027858][EUCADBEG] doDescribeInstances() invoked
[Mon Jun 27 15:22:06 2011][027858][EUCADBEG] doDescribeInstances(): instanceI
d=i-309406f5 publicIp=0.0.0.0 privateIp=10.10.100.130 mac=0610d3d1941061f5 via
n=10 networkIndex=2
[Mon Jun 27 15:22:12 2011][027858][EUCADBEG] doDescribeResource() invoked
[Mon Jun 27 15:22:12 2011][027858][EUCADBEG] doDescribeInstances() invoked
[Mon Jun 27 15:22:12 2011][027858][EUCADBEG] doDescribeInstances(): instanceI
d=i-309406f5 publicIp=0.0.0.0 privateIp=10.10.100.130 mac=0610d3d1941061f5 via
n=10 networkIndex=2
[Mon Jun 27 15:22:18 2011][027858][EUCADBEG] doDescribeResource() invoked
[Mon Jun 27 15:22:18 2011][027858][EUCADBEG] doDescribeInstances() invoked
[Mon Jun 27 15:22:18 2011][027858][EUCADBEG] doDescribeInstances(): instanceI
d=i-309406f5 publicIp=0.0.0.0 privateIp=10.10.100.130 mac=0610d3d1941061f5 via
n=10 networkIndex=2
[Mon Jun 27 15:22:24 2011][027858][EUCADBEG] doDescribeResource() invoked
[Mon Jun 27 15:22:24 2011][027858][EUCADBEG] doDescribeInstances() invoked
[Mon Jun 27 15:22:24 2011][027858][EUCADBEG] doDescribeInstances(): instanceI
d=i-309406f5 publicIp=0.0.0.0 privateIp=10.10.100.130 mac=0610d3d1941061f5 via
n=10 networkIndex=2
[Mon Jun 27 15:22:30 2011][027858][EUCADBEG] doDescribeResource() invoked
[Mon Jun 27 15:22:30 2011][027858][EUCADBEG] doDescribeInstances() invoked
[Mon Jun 27 15:22:30 2011][027858][EUCADBEG] doDescribeInstances(): instanceI
d=i-309406f5 publicIp=0.0.0.0 privateIp=10.10.100.130 mac=0610d3d1941061f5 via
n=10 networkIndex=2
```

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.

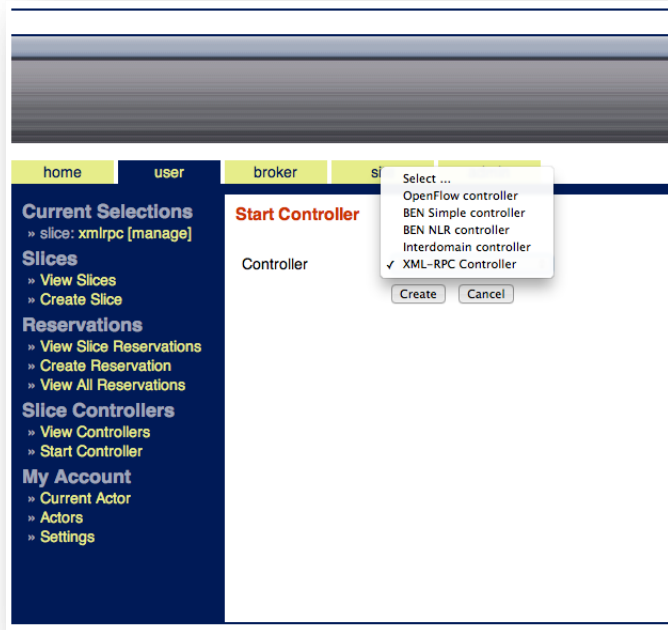


Figure 15. Starting an XML-RPC controller

Now, select “View Controllers”



Figure 16. View XML-RPC controller

Now select “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="25f53b64cfe44dd1604447f04b7b533bb67deale" />
      </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="25f53b64cfe44dd1604447f04b7b533bb67deale" />
        </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>

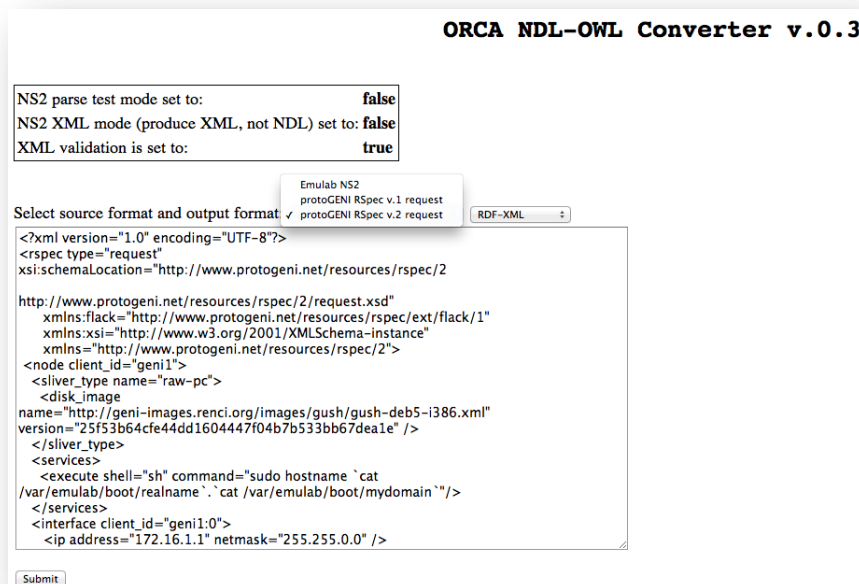


Figure 18. ORCA NDL-OWL Converter

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#25f53b64cfe44dd1604447f04b7b533bb67deale">
  <topology:hasURL>http://geni-images.renci.org/images/gush/gush-deb5-i386.xml</topology:hasURL>
  <topology:hasGUID>25f53b64cfe44dd1604447f04b7b533bb67deale</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#25f53b64cfe44dd1604447f04b7b533bb67deale"/>
  <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#25f53b64cfe44dd1604447f04b7b533bb67deale"/>
  <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#25f53b64cfe44dd1604447f04b7b533bb67deale"/>
  <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:
elcaff5b-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.”

No	Slice	Type	Units [R]	Units [A]	Start	End	Broker	Site	State	
1	xmlrpc	DUKE Internal EX3200	1	1	08/15/2011 09:59	08/16/2011 09:59	duke-broker1	duke-vm-site3	Active	manage
2	xmlrpc	Eucalyptus Virtual Machine (Duke)	1	1	08/15/2011 09:59	08/16/2011 09:59	duke-broker1	duke-vm-site3	Redeeming Ticket	manage
3	xmlrpc	Eucalyptus Virtual Machine (Duke)	1	1	08/15/2011 09:59	08/16/2011 09:59	duke-broker1	duke-vm-site3	Redeeming Ticket	manage

Figure 19. Manage XML-RPC instance

Reservation Details

Actions

Reservation ID b67e27e3-6c4f-4135-99d5-97a4dfb15d92

Resource Type Eucalyptus Virtual Machine (Duke)

Requested Units 1

Assigned Units 1

Leased Units 1

Lease Start 08/15/2011 11:40

Lease End 08/16/2011 11:40

Broker duke-broker1

Site duke-vm-site3

State Active

Notices Reservation b67e27e3-6c4f-4135-99d5-97a4dfb15d92 (Slice xmlrpc) is in state [Active,None]

No Properties

- unit.id=003bd703-c724-40a9-b416-beca1b58ec59
- shirako.save.unit.manage.port=22
- unit.manage.port=22
- unit.manage.ip=152.3.144.130
- unit.state=2
- unit.domain=dukevmsite
- shirako.save.unit.manage.ip=152.3.144.130
- unit.actorid=1d5616cc-4751-458d-acta-c367191df7ef
- shirako.save.unit.hostname.uri=\${unit.hostname.uri}
- unit.cpu=1.2
- shirako.save.unit.ec2.instanceid=5C2709DD
- unit.sliceid=999e184c-4950-42de-8694-5eb7e2661727
- unit.resourceType=duke.vm
- unit.sequence=2
- unit.id=b67e27e3-6c4f-4135-99d5-97a4dfb15d92
- unit.ec2.instanceid=5C2709DD
- unit.hostname.uri=\${unit.hostname.uri}
- unit.memory=1.7
- unit.ndi.adomain=10 http://geni-orca.renci.org/ow/dukevmsite.rdf true 100000000 100000000 eth1 http://geni-orca.renci.org/ow/dukevmsite.rdf ge-0/0/33-47

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
```

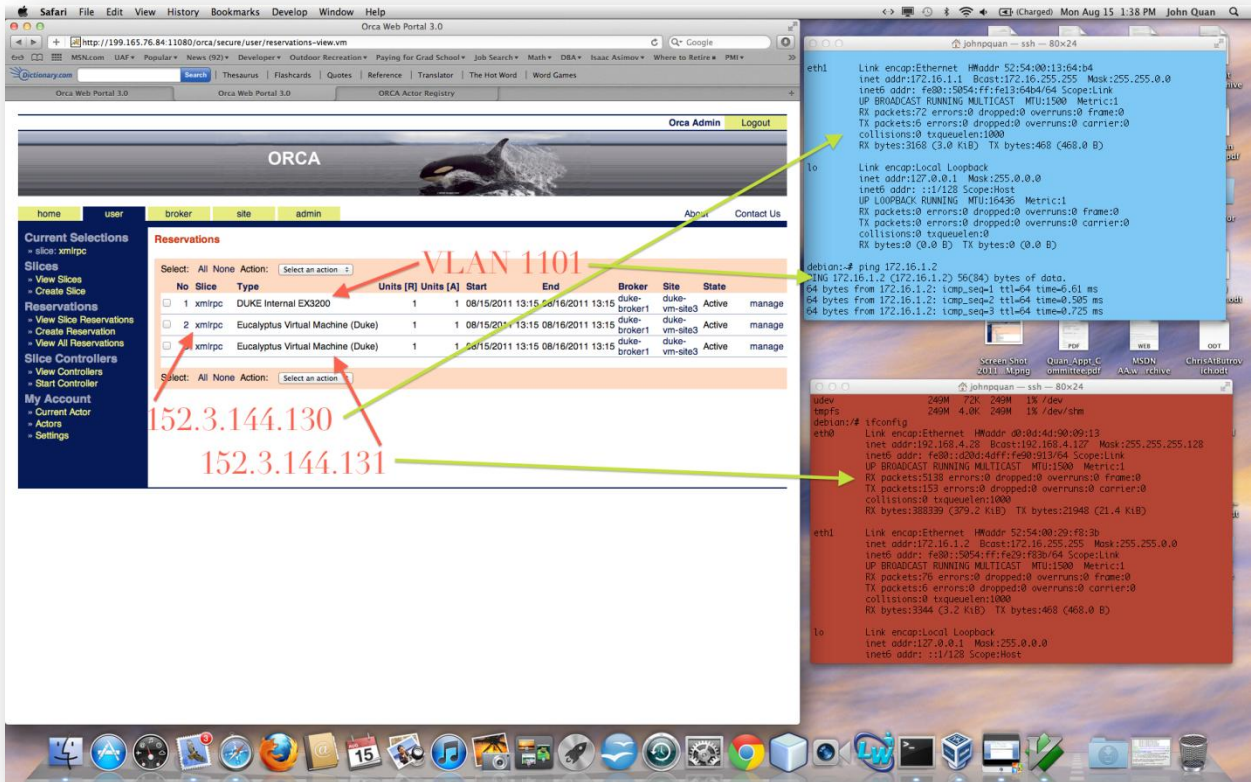


Figure 21. Simple XML-RPC slice

