Annex D – IST-124 EXPERIMENTATION EXECUTION

Kelvin Marcus
U.S. Army Research Laboratory (ARL)
UNITED STATES

NATO IST-124 RTG uses the US Army Research Laboratory's Dynamically Allocated Virtual Clustering Management System (DAVC) to deploy the Anglova scenario using the distributed EMANE emulation model [1]. In this emulation model the EMANE software is installed within VMs that execute the applications that are the subject of the experimentation and whose performance is being evaluated.

This annex is divided into 3 sections:

- Section D.1 details the steps required to launch the EMANE emulation of the IST-124-061 Anglova experimentation scenario within the DAVC environment.
- Section D.2 is a guide for DAVC system setup and configuration.
- Section D.3 is the DAVC user guide with step-by-step instructions to perform common DAVC operations to access and manage DAVC clusters, nodes, virtual hard disks, and persistent block storage.

D.1 EXPERIMENTATION EXECUTION

This portion of the annex provides guidance and instructions for executing the IST-124-RTG-061 experimentation environment. Specifically, it contains the instructions for executing the EMANE emulation for the 2nd and 3rd Anglova scenario vignettes. The first vignette has had the lowest priority in the group and is not fully modelled yet. For more information about the Anglova scenario and tools see:

- Annex A: "Operational Perspective for IST-124";
- Annex B "Emulation Based Experimentation and the Anglova Scenario"; and
- Annex C "Experimentation Environment and Tools".

D.1.1 Introduction

The IST-124-RTG-061 activity was focused on heterogeneous networks in the deployable and mobile tactical levels. A typical network can be illustrated with the scenario given in Figure D-1. This scenario was created to show the information needs and exemplify the challenges related to the heterogeneity of the network. The operational needs have been defined; tasks to be fulfilled, collaboration among organizational units, information management as well as communications, command and control systems used.

The scenario depicts an operation conducted by the company task forces of the mechanized battalion. They are part of the Military Contingent (MC) coordinated by the Coalition Head Quarter (HQ). The company Communications and Information System (CIS) is connected to the National Operational WAN and has access to the Coalition systems. The MC HQ plays the reach-back role during the operation and provides Combat Support (CS) and Combat Service Support (CSS) if requested. According to the operational context it is assumed that enemy's forces are preparing a complex attack against the coalition base from the village located in the operational zone. The enemies are well armoured and operate in an area which can be mined, so there is a chance of IED (Improvised Explosive Device) hazard. The task of the own forces is to move into the operational zone and neutralize the insurgents and to destroy the armaments they collected. It is very important to avoid village inhabitants' casualties and to make the insurgents' escape impossible. The most important elements in this mission are CIS, logistics and medical support, which are provided by Coalition Forces. A well-functioning communication platform to help organizing the armed forces is therefore required.

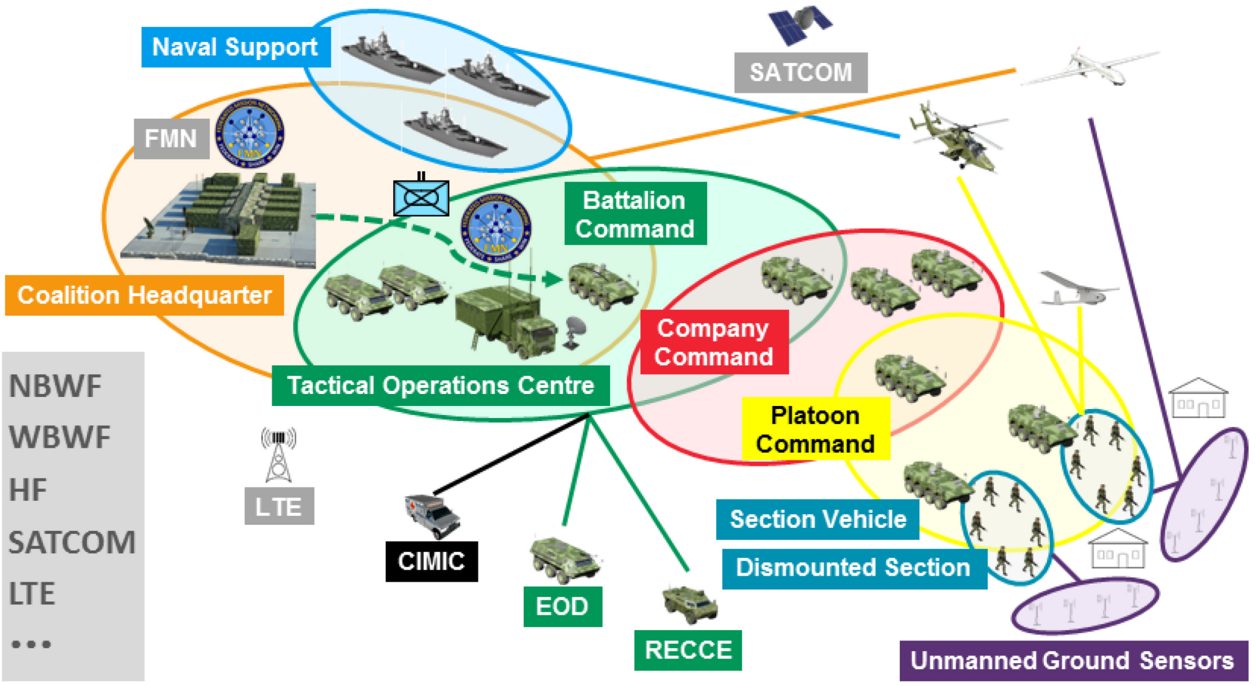

Figure D-1: Operational Scenario for IST-124 RTG-061.

The completion of the task mentioned requires access to a wide range of systems and communication networks, i.e., radio communications system (HF, VHF, UHF, SATCOM), sensor networks and Unmanned Aerial Vehicle (UAV) systems, NAVY management systems in terms of supporting reconnaissance and surveillance of the mission and services such as data, voice and video.

Three vignettes were defined in order to implement the actions included in the scenario. The roles and actors are the same for each vignette. The first vignette covers the intelligence preparation of the battlefield. The second vignette consists of the movement of coalition forces into the operational zone, including maritime interdiction operations in the surrounding coastal areas. The third vignette consists of an urban operation resulting in the neutralization of insurgents. The third vignette also includes a medical evacuation to a naval ship following the neutralization of IEDs. Each vignette describes data that are expected to be exchanged between the actors and C4ISR (Command, Control, Communications, and Computers Information System) equipment used in a way that emphasize the problems of connectivity and network efficiency of military heterogeneous networks.

D.1.2 Anglova Scenario

The Anglova scenario is a concrete realization of the operational context described above and depicts an operation conducted by a coalition task force including a maritime component. The tactical domain is located in the fictitious area of Fieldmont in Anglova, where the Coalition HQ (CHQ) of the Military Contingent (MC) is based. The scenario contains three vignettes as outlined in the operational context. However, the first vignette involving the Intelligence Preparation of the Battlefield has not yet been fully modelled.

The second vignette covers the deployment of the coalition forces, a battalion consisting of six companies, into the operational zone. The forces that are moving into the operational zone use a combination of narrowband VHF and wideband UHF connections for their own interoperability and operability with MC forces. The scenario includes detailed mobility patterns of the battalion north of Wellport in Anglova.

The battalion consists of six companies:

- Four mechanized companies with 24 vehicles each;
- One command and artillery company with 22 vehicles; and
- One support and supply company with 39 vehicles.

Together, there are 157 vehicles, each of them being a network node. The maritime component includes 21 ships and one multi-purpose helicopter that provides communications relays. In addition, a Coalition Headquarters node and an airborne node are also included in this vignette – a strategic UAV asset that can act as a communications relay and provide persistent surveillance capabilities.

The third vignette covers the urban counter-insurgency operation within the town of Wellport and involves three platoons (72 nodes), 10 unattended ground sensors, one aerial sensor (Aerostat), two UAVs (tactical and data harvest), three satellites, the 21 navy ships that are continuing the maritime mission, and the multi-purpose helicopter that is re-tasked for Medevac. The vignette is split into 3 parts which include neutralization of the insurgents and the IED, a Medevac from the urban environment to a naval ship, and the platoons returning to base.

The experimentation environment provides a common platform to explore research issues relevant to heterogeneous tactical networks including routing architectures and their impact on delivery rates, overheads, and scalability; data dissemination protocols; quality of service and resource management; and leveraging and integration of sensor networks. The instructions provided in this annex can be used as a guide to launch various subsets of the 269-node Anglova emulation scenario for a wide range of experimentation backdrops.

D.1.3 Experimentation Environment

The experimentation environment consists of the Dynamically Allocated Virtual Clustering (DAVC) management system, a customized Virtual Machine (VM) template preconfigured with the EMANE network emulator software, and scripting to launch vignettes 2 and 3 of the Anglova scenario.

D.1.3.1 Dynamically Allocated Virtual Clustering Management System (DAVC)

DAVC (Figure D-2) is a web based virtualization service and cloud-operating environment that creates complex virtual experimentation clusters that can be used for simulation-based, emulation-based, and hybrid field/emulation experimentation. DAVC deploys networked clusters composed of VMs tailored to user specifications. The DAVC management system abstracts away test-bed infrastructure configuration through automated provisioning processes that configure the virtual networking for each VM. Clusters created by DAVC are heterogeneous, so each VM can have different OSs, application sets, and hardware attributes such as RAM, CPU cores, hard disk, and network interfaces. DAVC users can register custom VMs as templates that can be used within their experimentation clusters.

Using DAVC, the Anglova scenario is distributed with a 1-to-1 mapping with each Anglova node running within a single DAVC virtual cluster node. The entire 269 node scenario can run within a 270 node DAVC cluster. When deployed in this manner node 270 acts as the experimentation orchestration node and is responsible for executing the bootstrap scripting that launches the various applications and EMANE on the remaining 269 nodes.

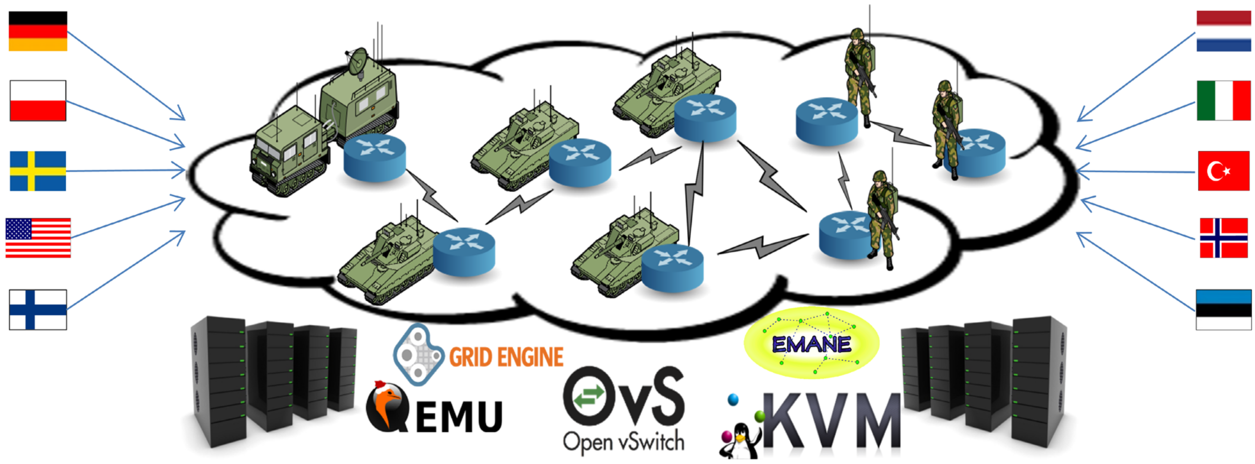

Figure D-2: Emulation Environment for IST-124, Available as a Cloud Service to all IST-124 Members.

D.1.3.2 Experimentation Virtual Machine

A custom Ubuntu 16.04 VM is used to represent a single Anglova scenario node. The template VM is preinstalled with the applications necessary for running the Anglova scenario including EMANE, the Multi Generator (MGEN) [2], and the Optimized Link State Routing Protocol (OLSR)v1 [3] and OLSRv2 [4] routing protocols. The VM is also preinstalled with the various EMANE radio models, mobility and path-loss configuration files specific to the Anglova scenario vignettes. Custom scripting to bootstrap the Anglova scenario and emulation environment is also preinstalled within the VM. This VM is registered with DAVC and used as a template within a DAVC experimentation cluster to run the Anglova scenario.

The specifics of the VM's file system including the EMANE configuration files and the experimentation scripting files will be covered in Section D.1.5. Section D.1.4 details the process to create a DAVC cluster that will host the VMs where the experimentation environment will be run.

D.1.4 DAVC Cluster Configuration

The first step in executing the experimentation environment is creating and launching a DAVC Cluster to host the VM nodes where the experimentation environment components will be run. This step will automatically provision multiple instances of the custom VM discussed in the previous section with the base settings necessary to run the EMANE emulation and the experimentation environment scripting. Specifically, this step will create a DAVC cluster consisting of 270 instances of the experimentation VM node connected to the following networks 172.15.0.0/23 and 172.16.0.0/23. The provisioned cluster will provide enough resources to run Anglova scenario Vignette 2 or all portions of Vignette 3.

D.1.4.1 Access and Logging into DAVC

Access the DAVC web application URL with a browser. On the DAVC home page, login using the username/password form at the top right of the application shown in Figure D-3.

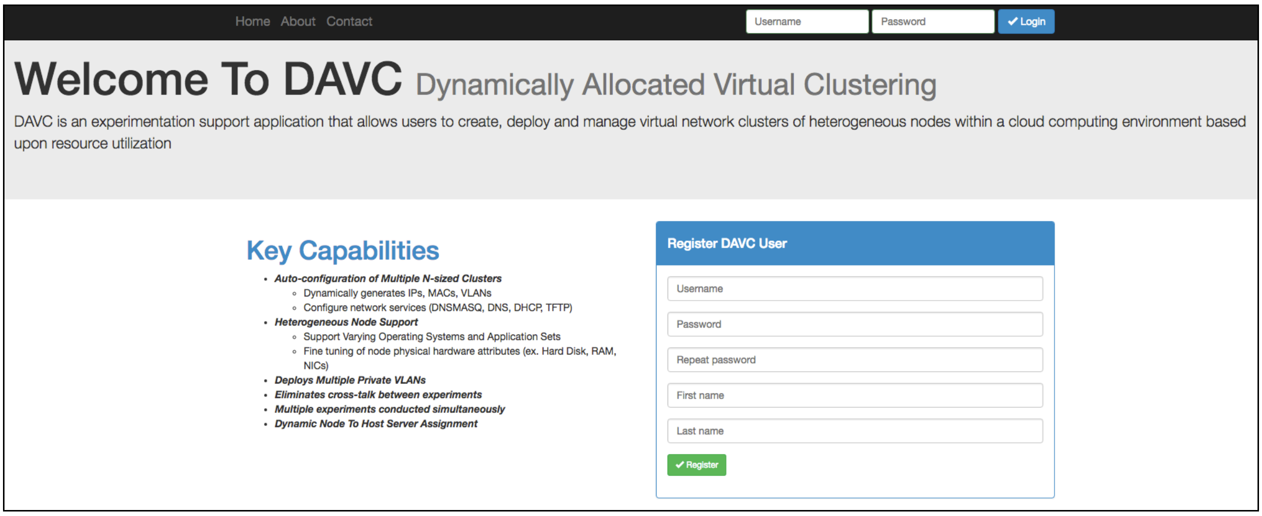

Figure D-3: DAVC Web Application Login.

D.1.4.2 Create the DAVC Cluster

On the user dashboard screen, shown in Figure D-4, press the "Create A Cluster" button to begin creating the 270 node DAVC cluster.

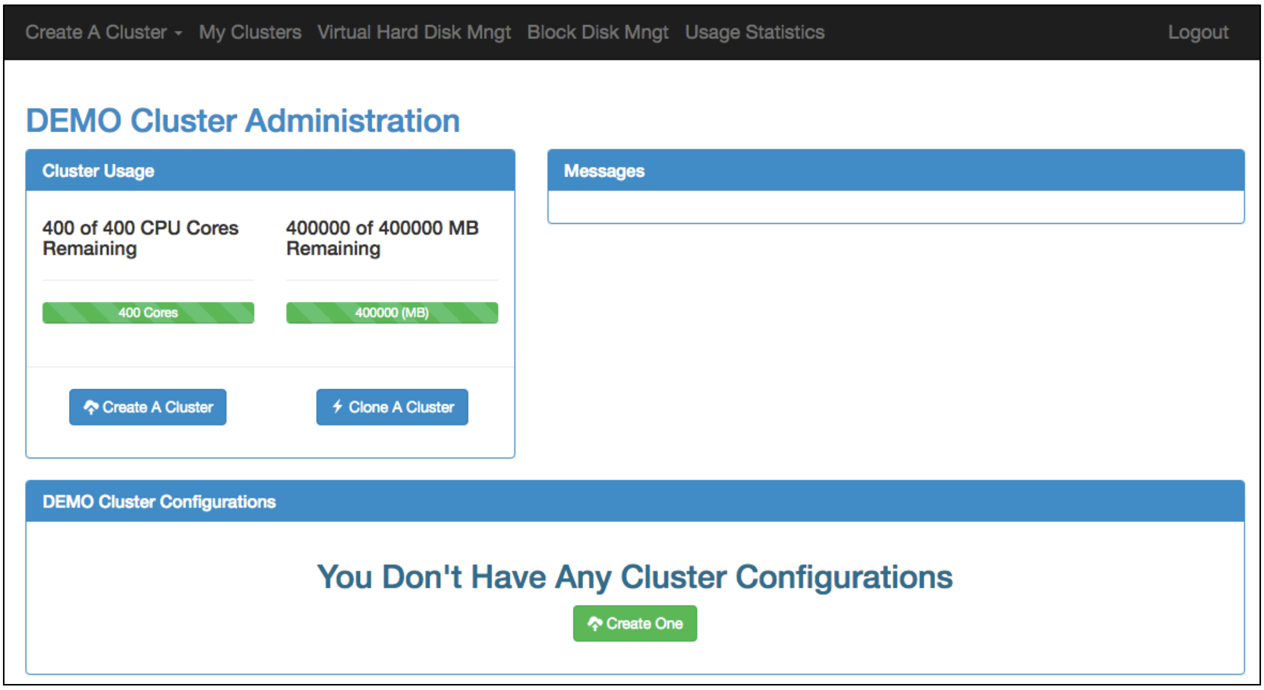

Figure D-4: DAVC User Dashboard.

Next, on the 'Create New DAVC Cluster' dialog's 'Cluster Info' tab (Figure D-5), set the cluster's name and description. Press the 'Create Networks' button when complete.

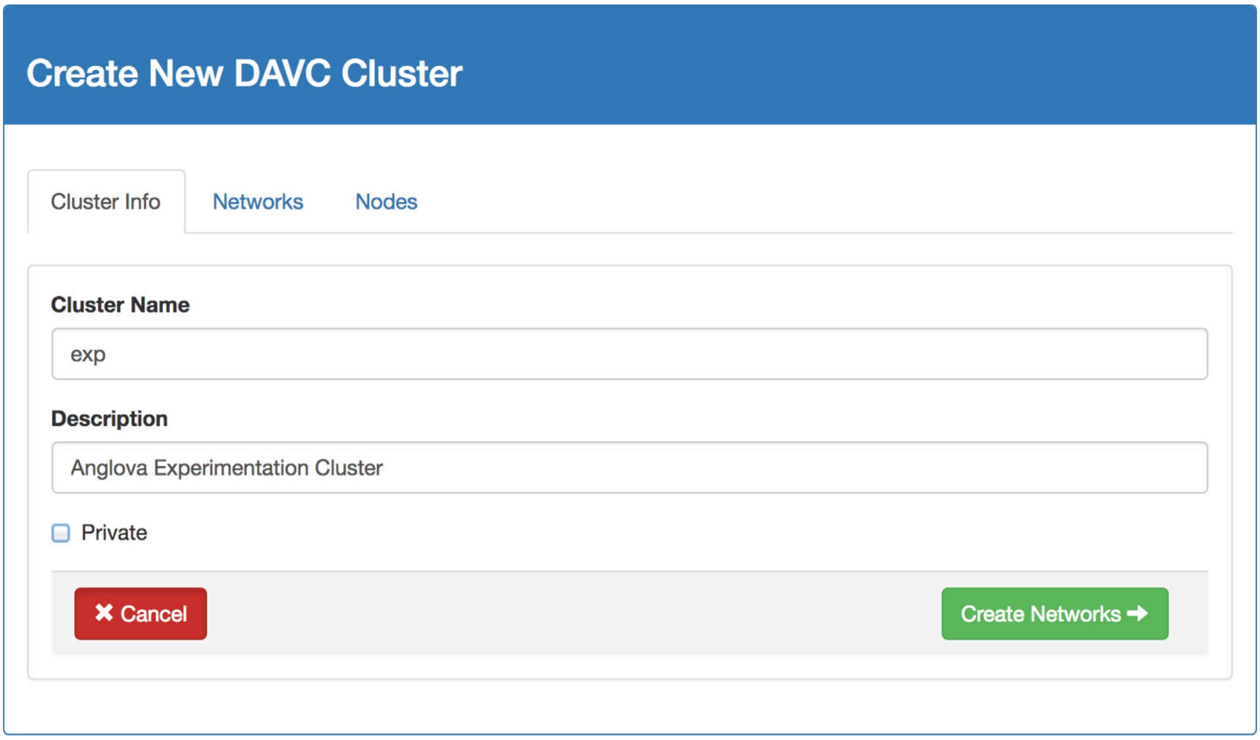

Figure D-5: Cluster Info Tab.

D.1.4.3 Create the DAVC Cluster Networks

On the ‘Networks’ tab (Figure D-6), press the “Add Cluster Networks” button to begin adding the two networks required (172.15.0.0/23 and 172.16.0.0/23) for the experimentation environment. These networks will be associated with the emulated EMANE ‘Over-The-Air’ radio network.

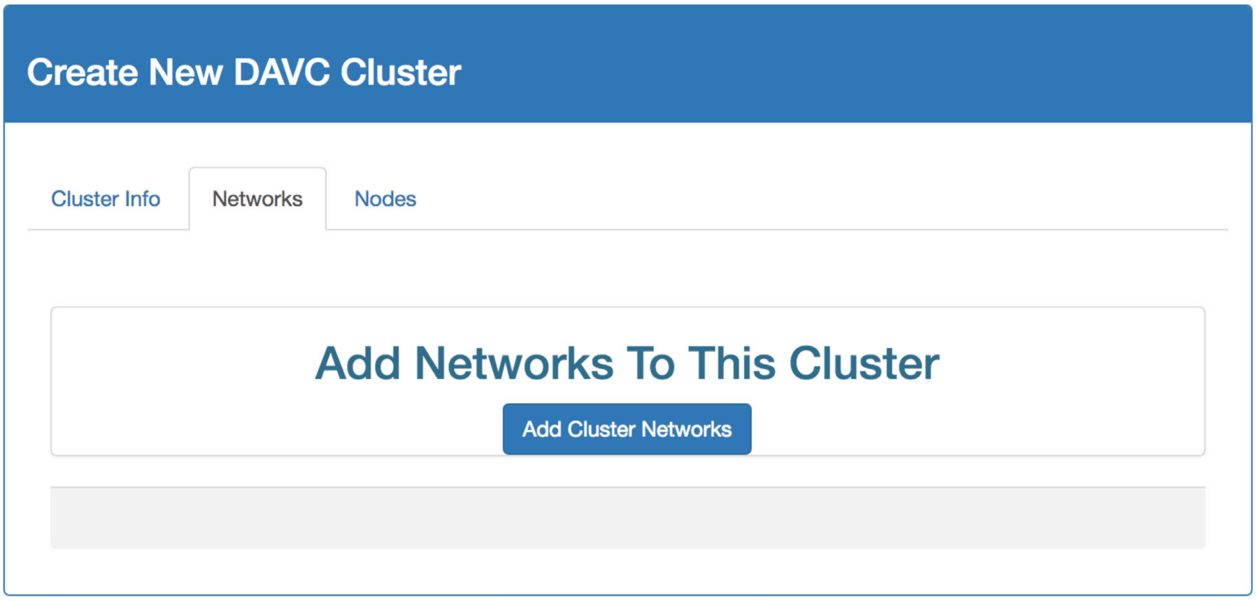

Figure D-6: Cluster Networks Tab.

Enter the network 172.15.0.0/23 into the 'Add a Network' dialog (Figure D-7) and press 'Add Network' when complete.

Figure D-7: Add the Network 172.15.0.0/23.

After adding the first network, press the "Add More Networks" button (Figure D-8) to add the second required network.

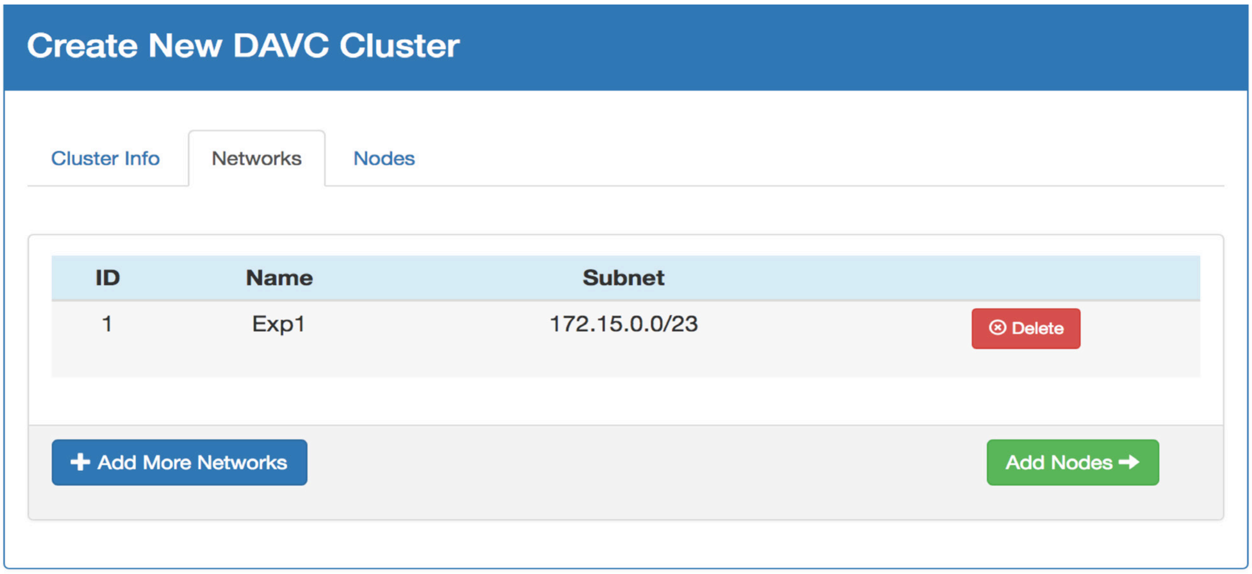

Figure D-8: Add the Second Network to the Cluster.

Enter the second required network 172.16.0.0/23 into the 'Add a Network' dialog (Figure D-9) and press 'Add Network' when complete.

Figure D-9: Add the Network 172.16.0.0/23.

After the two required networks have been set, press the 'Add Nodes' button (Figure D-10) to begin configuring the 270 VM nodes required for the experimentation environment.

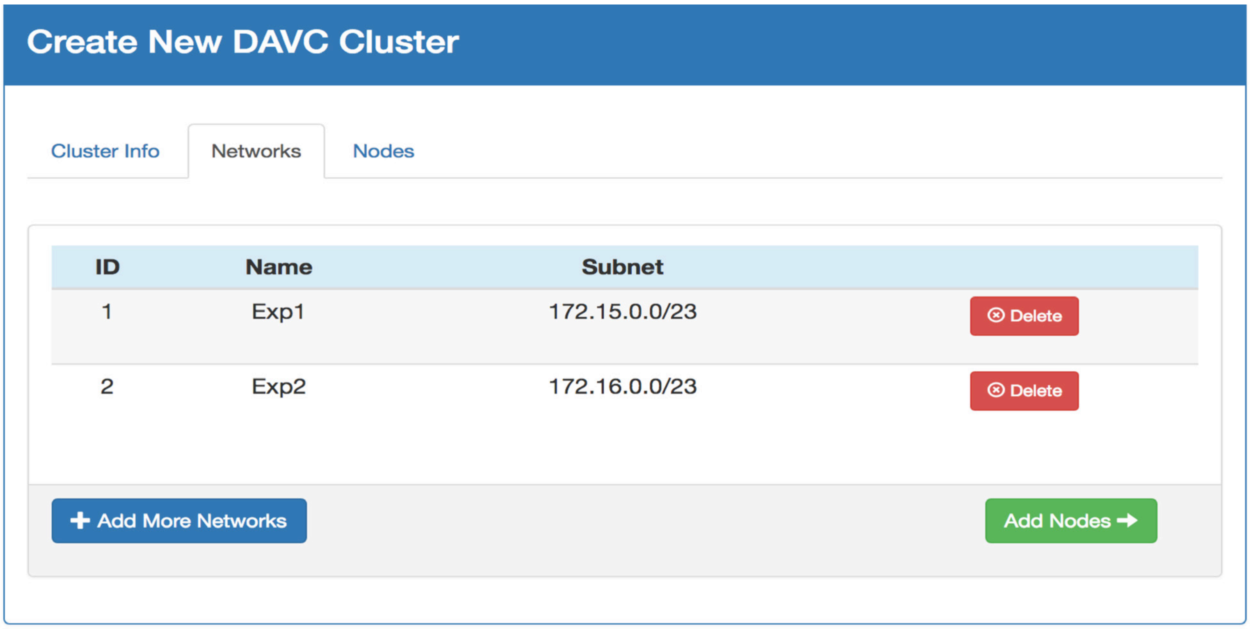

Figure D-10: The Two Experimentation Cluster Networks.

D.1.4.4 Create the DAVC Cluster Nodes

The experimentation environment DAVC cluster consists of a total of 270 VM nodes. VM nodes 1-269 are mapped to the various Anglova scenario nodes and VM node 270 is the experimentation controller node responsible for starting and stopping the experiment. On the 'Nodes' tab (Figure D-11), press the 'Add More Nodes' to begin configuring VM nodes 1-269.

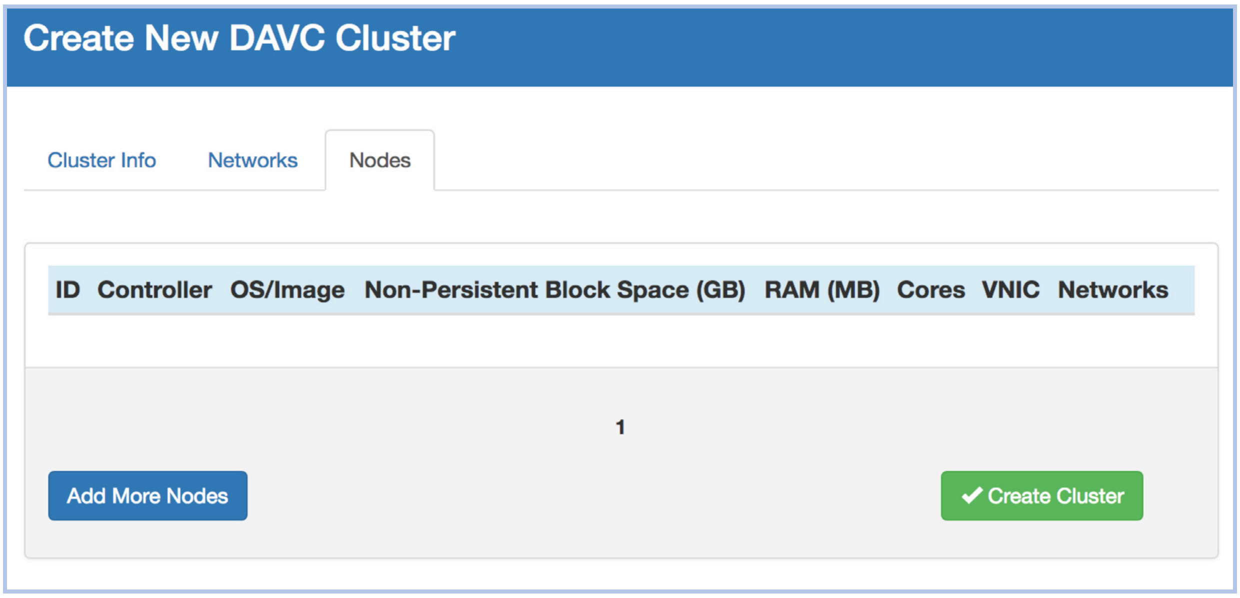

Figure D-11: Cluster Nodes Tab.

Configure VM nodes 1-269 as shown in Figure D-12. Each node uses the ‘Anglova_node_v3’ VM template and is configured with 1 CPU Core, 5GBs of non-persistent storage for logging, 2GBs of RAM, the virtio virtual network driver, and the 2 networks configured in Section D.1.4.3. Press the ‘Add Nodes’ button to add the 269 VM nodes.

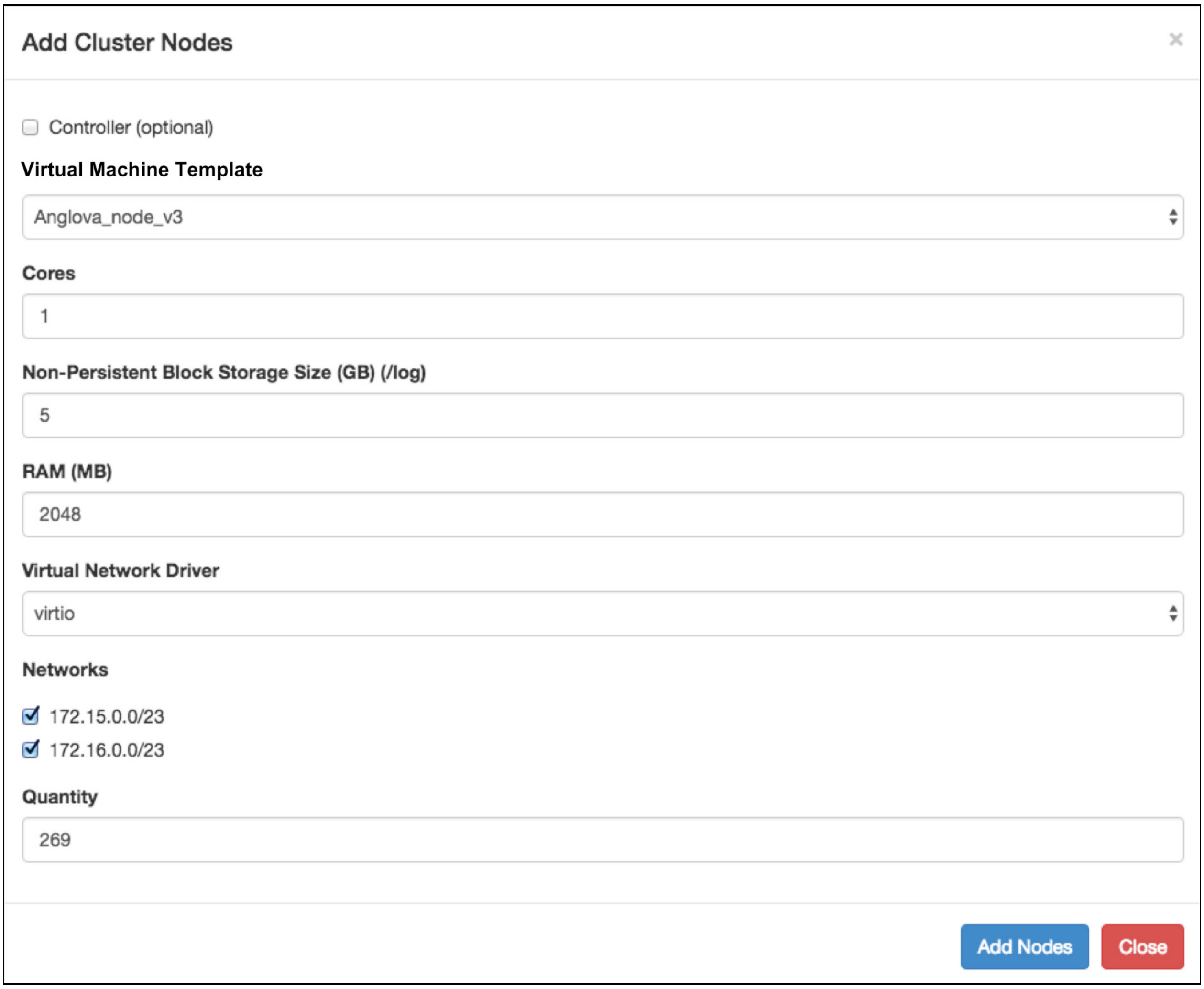

Figure D-12: Configure VM Nodes 1-269.

After adding VM nodes 1-269, press the ‘Add More Nodes’ button at the bottom of the cluster nodes dialog (Figure D-13) to configure and add the final DAVC cluster VM node.

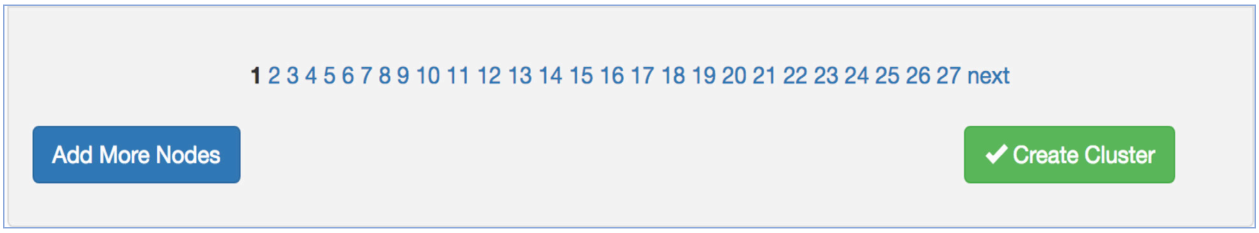

Figure D-13: Add VM Node 270.

Configure VM node 270 as shown in Figure D-14. This node also uses the 'Anglova_node_v3' VM template but it is configured with 6 CPU Core, 5GBs of non-persistent storage for logging, and 10GBs of RAM. It is also configured with the virtio virtual network driver, and the 2 networks configured in Section D.1.4.3. Press the 'Add Nodes' button to add this VM node to the cluster.

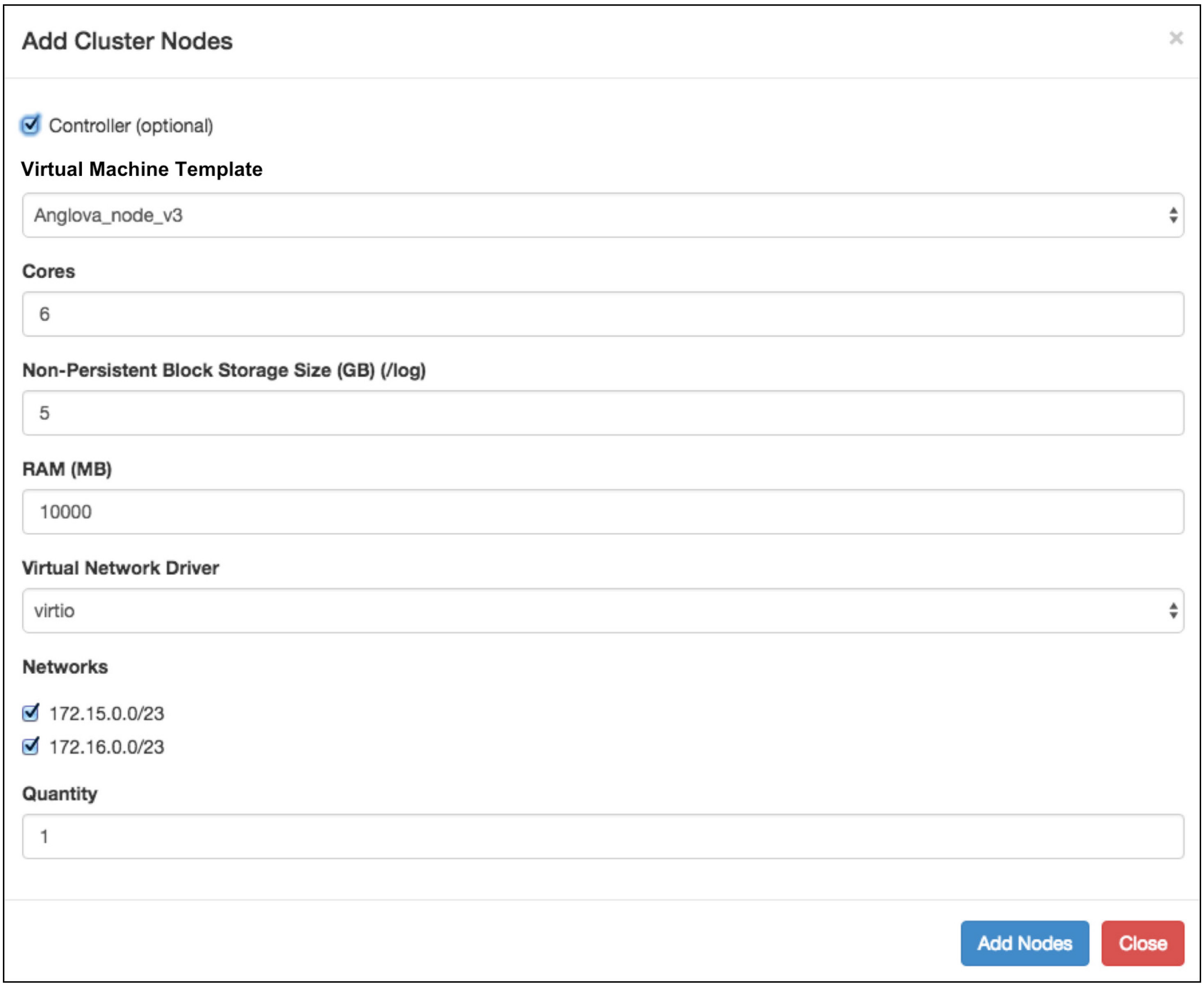

Figure D-14: Configure VM Node 270, the Experimentation Controller.

After all 270 VM nodes have been configured, press the 'Create Cluster' at the bottom of the cluster nodes dialog (Figure D-15) to complete the cluster creation process.

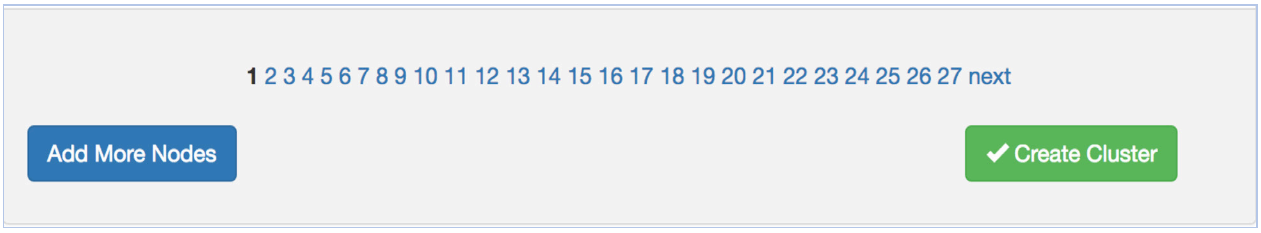

Figure D-15: Complete the Cluster Creation Process.

After completing the cluster creation process, the user is navigated to the cluster details page (Figure D-16) where the cluster can be launched.

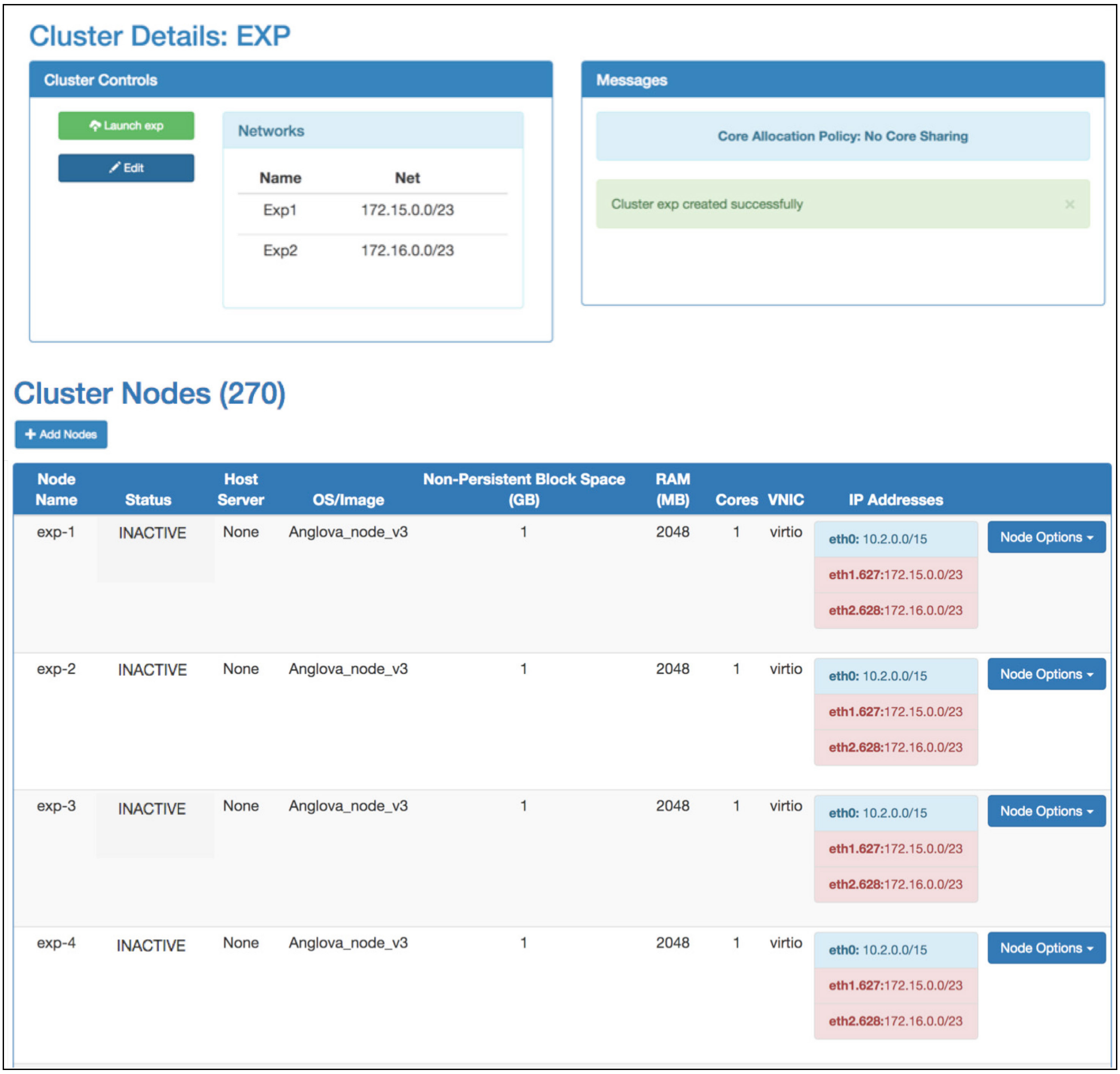

Figure D-16: Experimentation Cluster Created.

D.1.4.5 Launching the DAVC Cluster

Launch the cluster by pressing the green ‘Launch’ button in the ‘Cluster Controls’ box (Figure D-17). The status of the nodes will begin updating from ‘INACTIVE’ to ‘INITIALIZING’ and finally to ‘ACTIVE’ once the cluster is fully active. Up until this step the cluster has only been a configuration. Now the virtual machines are created, given resources from the servers and powered on.

The cluster activation process will take a while to fully complete as it involves copying and provisioning 270 VMs across several host servers. The activation process is complete and the cluster is active once all of the node’s status is updated to the green ‘ACTIVE’ label as shown in Figure D-18.

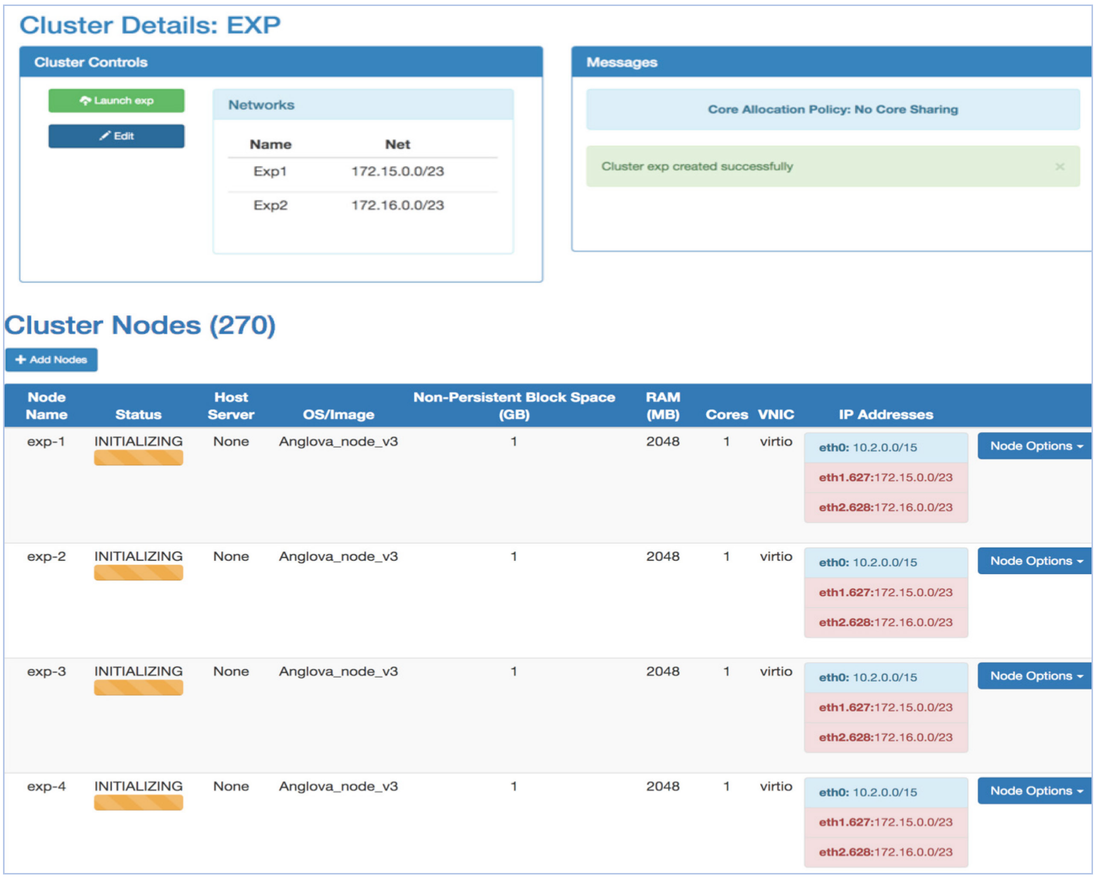

Figure D-17: Cluster Nodes Initializing.

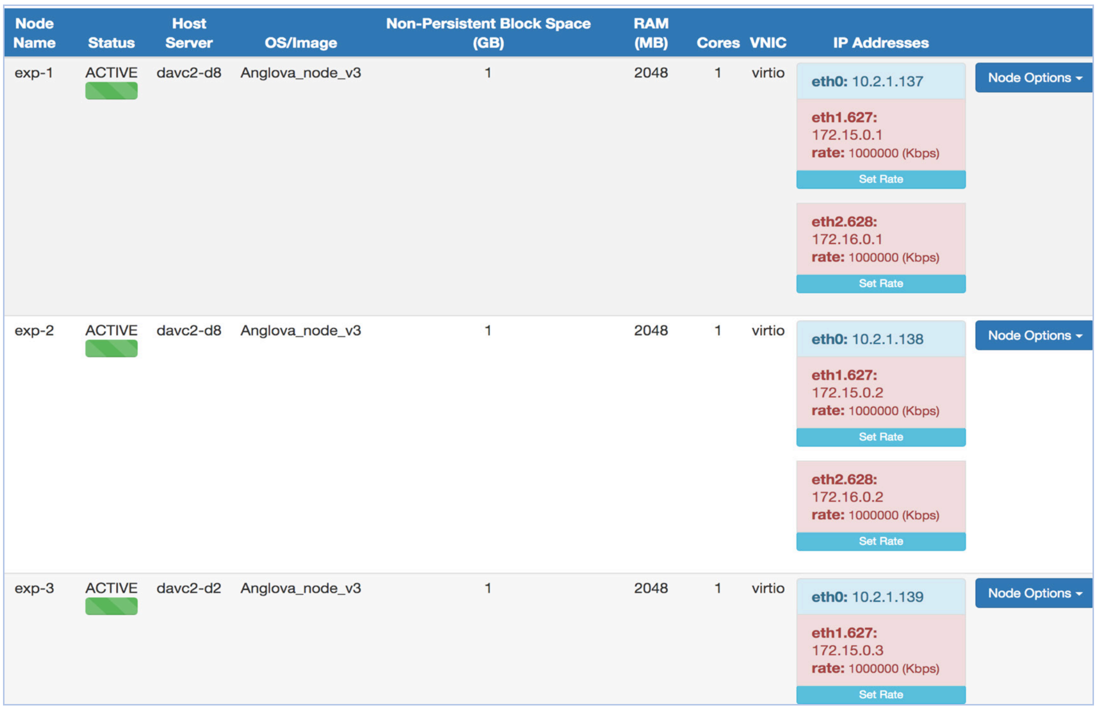

Figure D-18: The Cluster is Active Once All Nodes' Status is Set to 'ACTIVE'.

D.1.4.6 Logging into the Experimentation Controller

When the status of all of the nodes in the experimentation cluster is marked active, log into VM node 270's Virtual Network Computing (VNC) console by clicking on its 'Open VNC' button in the 'Node Options' dropdown menu (Figure D-19).

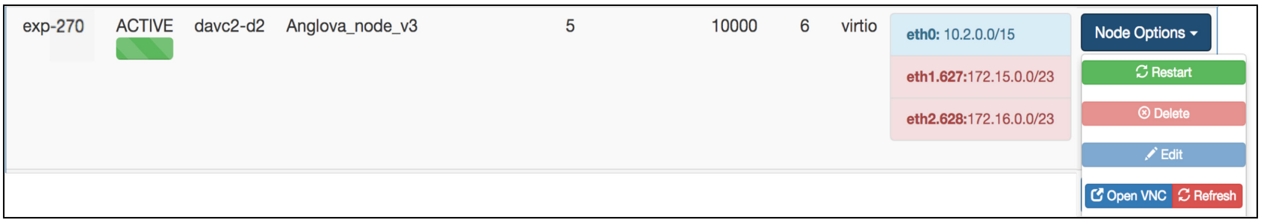

Figure D-19: Log into VM Node 270's VNC Console.

This will open a browser page with a VNC session hosting VM Node 270's desktop (Figure D-20). This node will run the experimentation scripting, which bootstraps the other cluster nodes to run the appropriate EMANE configurations and scripts for the specified vignette (Vignette 2 or Vignette 3 Part 1, Part 2, or Part 3). VM Node 270 will also run the EMANE event service, which will generate the location and path loss data for the specified vignette.

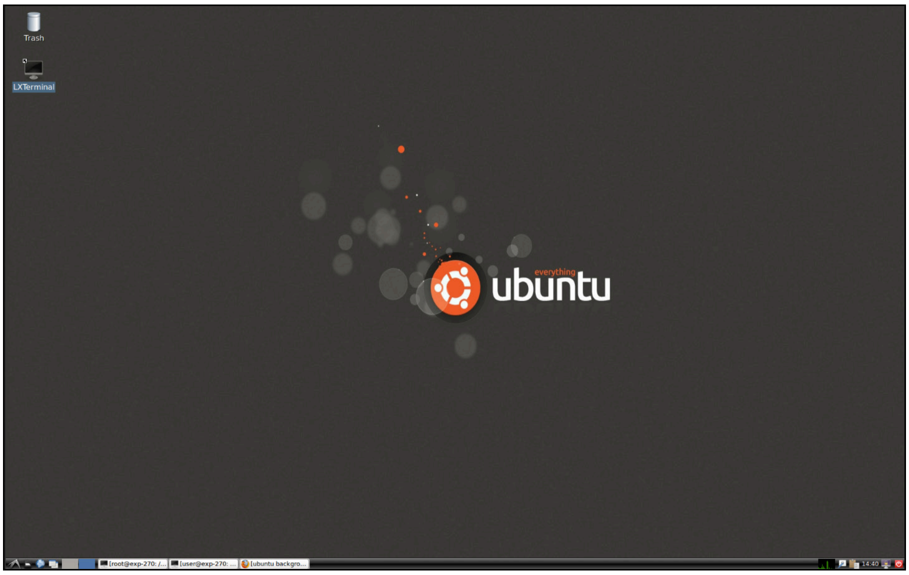

Figure D-20: VM Node 270's VNC Console.

The experimentation environment's DAVC cluster is ready and the Anglova scenario emulation can now be run. But first the experimentation environment's file system including the EMANE configuration files and the experimentation scripting files are described in the next section.

D.1.5 Experimentation Virtual Machine File System

A custom Ubuntu 16.04 VM is used to represent a single Anglova scenario node. The template VM is preinstalled with the applications necessary for running the Anglova scenario including EMANE, the Multi Generator (MGEN), and the OLSRv1 and OLSRv2 routing protocols. The VM is also preinstalled with the various EMANE radio models, mobility and path-loss configuration files specific to the Anglova scenario vignettes. Custom scripting to bootstrap the Anglova scenario and emulation environment is also preinstalled within the VM. As outlined in the previous section, this VM is used as the template VM to create a 270 node DAVC experimentation cluster to run the Anglova scenario.

The experimentation environment's file system including the EMANE configuration files, and the experimentation scripting files are described in the following sections.

D.1.5.1 Experimentation Configuration Files

All of the experimentation environment's configuration and scripting files are located in the /opt/nato-experiment directory shown in Figure D-21.

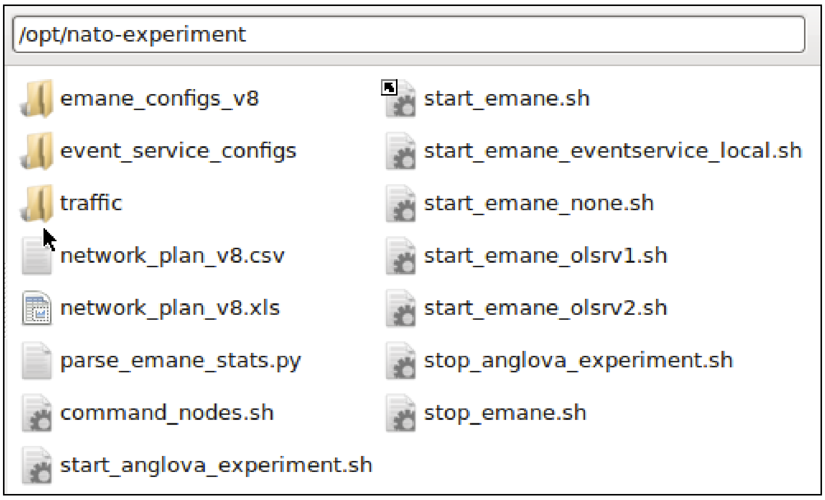

Figure D-21: Emulation Environment File System.

Table D-1 provides a brief description of the files located in this directory. A more detail description of these files will be covered in later sections.

Table D-1: A Brief Description of the Files Located in the /opt/nato-experiment Folder of the Controller Node.

File	Description
start_anglova_experiment.sh	Starts the experiment components on the DAVC cluster nodes.
stop_anglova_experiment.sh	Stops the experiment components on the DAVC cluster nodes.
start_emane.sh	Symbolic link that points to one of the other start_emane_<routing protocol version>.sh scripts.
start_emane_none.sh	Script to start the EMANE emulator without a routing protocol on a DAVC cluster node.
start_emane_olsrv1.sh	Script to start the EMANE emulator with the OLSRv1 routing protocol on a DAVC cluster node.
start_emane_olsrv2.sh	Script to start the EMANE emulator with the OLSRv2 routing protocol on a DAVC cluster node.
stop_emane.sh	Script to stop the EMANE emulator running on the DAVC cluster nodes.

File	Description
start_emane_eventservice.sh	Script to start the EMANE event service, which sends location and path loss events to the event service daemons on the DAVC cluster nodes.
parse_emane_stats.sh	Script to output the statistics and values retrieved from the EMANE shell.
command_nodes.sh	Utility script used to run a command on all of the DAVC cluster nodes
emane_configs_v8	Directory: Contains the EMANE platform and radio configuration files for the nodes in the Anglova emulation.
network_plan_v8.xls	Network plan spreadsheets containing node to network mappings for the Anglova emulation.
network_plan_v8.csv	Comma separated version of the network_plan_v8.xls file.
event_service_configs	Directory: Contains the EMANE mobility and path loss configuration files used to generate location and path loss events.
traffic	Directory: Contains the MGEN configurations and scripting to generate background traffic.

D.1.5.2 Experimentation Environment Network Plan

The experimentation environment's network plan (*/opt/nato-experiment/network_plan_v8.xls*) is a set of spreadsheets used by the experimentation scripting to configure the correct settings for each node's EMANE Network Emulation Module (NEM).

A portion of the experimentation environment's network plan is shown in Figure D-22. The network plan defines how the Anglova scenario nodes are mapped to the 17 emulated EMANE radio networks (See Annex B for more information about the modelling of the radios). Specifically it shows the node groupings (column 2) and their membership within the radio networks (rows 1 and 2, columns 3 to 17). A green entry in a column indicates that node is a member of and has a radio on the corresponding network. The numbers in the green cells are the radio IDs that will be assigned to the emulated EMANE radio. A dark gray '0' entry indicates the node does not have a radio on the corresponding network.

id	group	wideband1	wideband2	wideband3	wideband4	narrowband1	narrowband2	narrowband3	satcom	uav	iridium	cellular	tacsat-hf	ugs-net	navy-uhf	navy-hf	aerostat-fiber	aerostat-rf
network	subnets	192.168.1.0	192.168.2.0	192.168.3.0	192.168.4.0	192.168.5.0	192.168.6.0	192.168.7.0	192.168.8.0	192.168.9.0	192.168.10.0	192.168.11.0	192.168.12.0	192.168.13.0	192.168.14.0	192.168.15.0	192.168.16.0	192.168.17.0
1	company1	1	0	0	0	50	0	0	0	0	0	0	0	0	0	0	0	0
2	company1	100	0	0	0	150	0	0	0	0	0	0	0	0	0	0	0	0
3	company1	200	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	company1	250	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	company1	300	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	company1	350	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	company1	400	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	company1	450	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	company1	500	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	company1	550	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	company1	600	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	company1	650	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	company1	700	0	0	0	0	0	0	750	0	0	0	0	0	0	0	0	0
14	company1	800	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	company1	850	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	company1	900	0	0	0	0	0	0	950	0	0	0	0	0	0	0	0	0
17	company1	1000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	company1	1050	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	company1	1100	0	0	0	0	0	0	0	1150	0	0	0	0	0	0	0	0
20	company1	1200	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	company1	1250	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	company1	1300	0	0	0	0	0	0	0	1350	0	0	0	0	0	0	0	0
23	company1	1400	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	company1	1450	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	company2	0	1500	0	0	1550	0	0	0	0	0	0	0	0	0	0	0	0
26	company2	0	2000	0	0	2050	0	0	0	0	0	0	0	0	0	0	0	0
27	company2	0	2100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	company2	0	2150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	company2	0	2200	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	company2	0	2250	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	company2	0	2300	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	company2	0	2350	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	company2	0	2400	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	company2	0	2450	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	company2	0	2500	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	company2	0	2550	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	company2	0	2600	0	0	0	0	0	2650	0	0	0	0	0	0	0	0	0
38	company2	0	2700	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39	company2	0	2750	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	company2	0	2800	0	0	0	0	0	2850	0	0	0	0	0	0	0	0	0
41	company2	0	2950	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
42	company2	0	3000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
43	company2	0	3050	0	0	0	0	0	0	3100	0	0	0	0	0	0	0	0
44	company2	0	3150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
45	company2	0	3200	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
46	company2	0	3250	0	0	0	0	0	0	3300	0	0	0	0	0	0	0	0
47	company2	0	3350	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
48	company2	0	3400	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure D-22: Snippet of the Anglova Scenario Network Plan.

Taking a closer look at the network plan we can see in Figure D-23 that Anglova scenario nodes 1 and 2 are members of the ‘company1’ group and have radios on the ‘wideband1’ and ‘narrowband1’ networks. In this documentation an Anglova scenario node will be referred by its group name and position within that group. Using this convention we refer to Anglova scenario nodes 1 and 2 as company1-1 and company1-2 respectively. Note that Anglova scenario node 25 is referred to as company2-1 not company2-25. The IDs for the radios on company1-1 are 1 and 50. The IDs for the radios on company1-2 are 100 and 150. The ‘wideband1’ and ‘narrowband1’ network subnets are ‘192.168.1.0/24’ and ‘192.168.5.0/24’ respectively. When the emulation is started, the VM nodes mapped to the Anglova scenario nodes company1-1 and company1-2 will have EMANE radios configured on these subnets. The IP address assignments for the subnets are sequential per radio and across groups as shown in the network plan snippet in Figure D-24. Each radio is also assigned a host name in the format *<group>-<group id>-<radio name>*. Examples of the host name assignments are shown in the network plan snippet in Figure D-25.

id	group	wideband1	wideband2	wideband3	wideband4	narrowband1	narrowband2	narrowband3	satcom	uav
network	subnets	192.168.1.0	192.168.2.0	192.168.3.0	192.168.4.0	192.168.5.0	192.168.6.0	192.168.7.0	192.168.8.0	192.168.9.0
1	company1	1	0	0	0	50	0	0	0	0
2	company1	100	0	0	0	150	0	0	0	0

Figure D-23: Company1-1 and Company1-2 Network Mappings.

id	group	wideband1	wideband2	wideband3	wideband4	narrowband1	narrowband2	narrowband3	satcom	uav
network	subnets	192.168.1.0	192.168.2.0	192.168.3.0	192.168.4.0	192.168.5.0	192.168.6.0	192.168.7.0	192.168.8.0	192.168.9.0
1	company1	192.168.1.1	0	0	0	192.168.5.1	0	0	0	0
2	company1	192.168.1.2	0	0	0	192.168.5.2	0	0	0	0
3	company1	192.168.1.3	0	0	0	0	0	0	0	0
4	company1	192.168.1.4	0	0	0	0	0	0	0	0
5	company1	192.168.1.5	0	0	0	0	0	0	0	0
6	company1	192.168.1.6	0	0	0	0	0	0	0	0
7	company1	192.168.1.7	0	0	0	0	0	0	0	0
8	company1	192.168.1.8	0	0	0	0	0	0	0	0
9	company1	192.168.1.9	0	0	0	0	0	0	0	0
10	company1	192.168.1.10	0	0	0	0	0	0	0	0
11	company1	192.168.1.11	0	0	0	0	0	0	0	0
12	company1	192.168.1.12	0	0	0	0	0	0	0	0
13	company1	192.168.1.13	0	0	0	0	0	0	192.168.8.1	0
14	company1	192.168.1.14	0	0	0	0	0	0	0	0
15	company1	192.168.1.15	0	0	0	0	0	0	0	0
16	company1	192.168.1.16	0	0	0	0	0	0	192.168.8.2	0
17	company1	192.168.1.17	0	0	0	0	0	0	0	0
18	company1	192.168.1.18	0	0	0	0	0	0	0	0
19	company1	192.168.1.19	0	0	0	0	0	0	0	192.168.9.1
20	company1	192.168.1.20	0	0	0	0	0	0	0	0
21	company1	192.168.1.21	0	0	0	0	0	0	0	0
22	company1	192.168.1.22	0	0	0	0	0	0	0	192.168.9.2
23	company1	192.168.1.23	0	0	0	0	0	0	0	0
24	company1	192.168.1.24	0	0	0	0	0	0	0	0
25	company2	0	192.168.2.1	0	0	192.168.5.3	0	0	0	0
26	company2	0	192.168.2.2	0	0	192.168.5.4	0	0	0	0
27	company2	0	192.168.2.3	0	0	0	0	0	0	0
28	company2	0	192.168.2.4	0	0	0	0	0	0	0
29	company2	0	192.168.2.5	0	0	0	0	0	0	0
30	company2	0	192.168.2.6	0	0	0	0	0	0	0
31	company2	0	192.168.2.7	0	0	0	0	0	0	0
32	company2	0	192.168.2.8	0	0	0	0	0	0	0
33	company2	0	192.168.2.9	0	0	0	0	0	0	0
34	company2	0	192.168.2.10	0	0	0	0	0	0	0
35	company2	0	192.168.2.11	0	0	0	0	0	0	0
36	company2	0	192.168.2.12	0	0	0	0	0	0	0
37	company2	0	192.168.2.13	0	0	0	0	0	192.168.8.3	0
38	company2	0	192.168.2.14	0	0	0	0	0	0	0
39	company2	0	192.168.2.15	0	0	0	0	0	0	0
40	company2	0	192.168.2.16	0	0	0	0	0	192.168.8.4	0
41	company2	0	192.168.2.17	0	0	0	0	0	0	0

Figure D-24: Network Plan IP Address Mappings.

According to the network plan, Anglova scenario nodes company1-1 and company1-2 will be assigned the following radios (Table D-2).

Note that not all nodes are involved in each vignette. The network plan also includes a spreadsheet that specifies which groups are mapped to each vignette. The group vignette mappings are shown in Figure D-26.

Table D-2: The Radio Networks of Company 1.

company1-1	Host name: company1-1-wideband1 IP Address: 192.168.1.1/24	Host name: company1-1-narrowband1 IP Address: 192.168.5.1/24
company1-2	Host name: company1-2-wideband1 IP Address: 192.168.2.2/24	Host name: company1-2-narrowband1 IP Address: 192.168.5.2/24

id	group	wideband1	wideband2	wideband3	wideband4	narrowband1	narrowband2	narrowband3	satcom	uav
network	subnets	192.168.1.0	192.168.2.0	192.168.3.0	192.168.4.0	192.168.5.0	192.168.6.0	192.168.7.0	192.168.8.0	192.168.9.0
1	company1	company1-1-wideband1	0	0	0	company1-1-narrowband1	0	0	0	0
2	company1	company1-2-wideband1	0	0	0	company1-2-narrowband1	0	0	0	0
3	company1	company1-3-wideband1	0	0	0	0	0	0	0	0
4	company1	company1-4-wideband1	0	0	0	0	0	0	0	0
5	company1	company1-5-wideband1	0	0	0	0	0	0	0	0
6	company1	company1-6-wideband1	0	0	0	0	0	0	0	0
7	company1	company1-7-wideband1	0	0	0	0	0	0	0	0
8	company1	company1-8-wideband1	0	0	0	0	0	0	0	0
9	company1	company1-9-wideband1	0	0	0	0	0	0	0	0
10	company1	company1-10-wideband1	0	0	0	0	0	0	0	0
11	company1	company1-11-wideband1	0	0	0	0	0	0	0	0
12	company1	company1-12-wideband1	0	0	0	0	0	0	0	0
13	company1	company1-13-wideband1	0	0	0	0	0	0	company1-13-satcom	0
14	company1	company1-14-wideband1	0	0	0	0	0	0	0	0
15	company1	company1-15-wideband1	0	0	0	0	0	0	0	0
16	company1	company1-16-wideband1	0	0	0	0	0	0	company1-16-satcom	0
17	company1	company1-17-wideband1	0	0	0	0	0	0	0	0
18	company1	company1-18-wideband1	0	0	0	0	0	0	0	0
19	company1	company1-19-wideband1	0	0	0	0	0	0	0	company1-19-uav
20	company1	company1-20-wideband1	0	0	0	0	0	0	0	0
21	company1	company1-21-wideband1	0	0	0	0	0	0	0	0
22	company1	company1-22-wideband1	0	0	0	0	0	0	0	company1-22-uav
23	company1	company1-23-wideband1	0	0	0	0	0	0	0	0
24	company1	company1-24-wideband1	0	0	0	0	0	0	0	0
25	company2	0	company2-1-wideband2	0	0	company2-1-narrowband1	0	0	0	0
26	company2	0	company2-2-wideband2	0	0	company2-2-narrowband1	0	0	0	0
27	company2	0	company2-3-wideband2	0	0	0	0	0	0	0
28	company2	0	company2-4-wideband2	0	0	0	0	0	0	0
29	company2	0	company2-5-wideband2	0	0	0	0	0	0	0
30	company2	0	company2-6-wideband2	0	0	0	0	0	0	0
31	company2	0	company2-7-wideband2	0	0	0	0	0	0	0
32	company2	0	company2-8-wideband2	0	0	0	0	0	0	0
33	company2	0	company2-9-wideband2	0	0	0	0	0	0	0
34	company2	0	company2-10-wideband2	0	0	0	0	0	0	0
35	company2	0	company2-11-wideband2	0	0	0	0	0	0	0
36	company2	0	company2-12-wideband2	0	0	0	0	0	0	0
37	company2	0	company2-13-wideband2	0	0	0	0	0	company2-13-satcom	0
38	company2	0	company2-14-wideband2	0	0	0	0	0	0	0
39	company2	0	company2-15-wideband2	0	0	0	0	0	0	0
40	company2	0	company2-16-wideband2	0	0	0	0	0	company2-16-satcom	0
41	company2	0	company2-17-wideband2	0	0	0	0	0	0	0

Figure D-25: Network Plan Host Names.

group	Vignette 2	Vignette 3-1	Vignette 3-2	Vignette 3-3
company1	*			
company2	*			
company3	*			
company4	*			
command	*			
support1	*			
support2	*			
platoon1		*	*	*
platoon2		*	*	*
platoon3		*	*	*
tac-uav	*	*	*	*
harvest-uav		*	*	*
navy		*	*	*
med-helicopter		*	*	*
aerostat		*	*	*
tac-sat		*	*	*
iridium-sat		*	*	*
geo-sat		*	*	*
ugs		*	*	*
coalition-hq	*	*	*	*
toc-hq	*	*	*	*

Figure D-26: Vignette Group Mappings.

D.1.5.3 EMANE Configuration Files

The EMANE configuration and radio model files for running the experimentation environment are separated by group in the */opt/nato-experiment/emane_configs_v8* directory shown in Figure D-27.

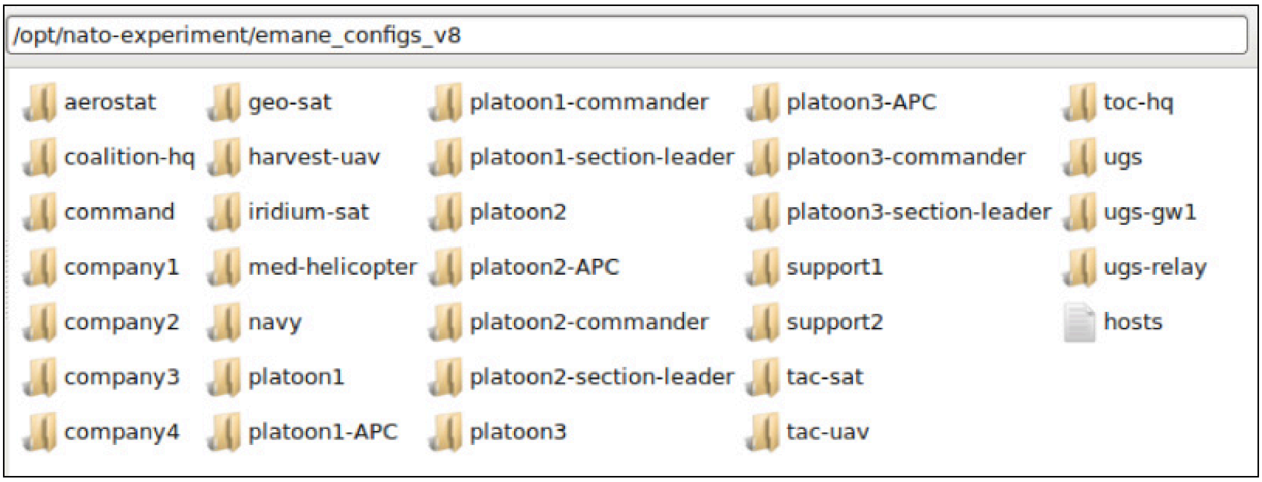

Figure D-27: EMANE Configuration Directories.

Each group directory contains the EMANE configuration files required to instantiate the EMANE radios for each Anglova scenario node within that particular group as defined by the network plan discussed in Section D.1.5.2. EMANE requires several Extensible Markup Language (XML) configuration files to properly instantiate a node's emulated radio. These include files listed in Table D-3.

Table D-3: The Set of Xml Files that are Used to Instantiate a Node's Emulated Radio.

File	Description
platform.xml	Defines radio model instantiations and their multicast group/interface mappings. One file per node.
eventdaemon.xml	Defines the multicast group/interface mappings for the event daemon. One file per emulated radio.
gpsdlocationagent.xml	Defines the configuration parameters for the GPS Daemon (GPSD) location agent. One file per emulated radio.
transvirtual.xml	Defines the transport component responsible for delivering messages between an emulator instance and application space processes. One per node.
radio_nem.xml	Specifies the radio model's Medium Access (MAC) and Physical layer (PHY) configuration files. One per radio network.
radio_mac.xml	Specifies the radio model's MAC layer configurations. One per radio network.
radio_phy.xml	Specifics the radio model's PHY layer configurations. One per radio network.

Figure D-28 shows the EMANE configuration files for the ‘company1’ group. Since the ‘company1’ group consists of 24 nodes there are 24 different ‘platform.xml’ files, one per node. The naming convention for the platform files is ‘platform<platform ID>.xml’, where the platform ID is the ID from column 1 of the network plan file discussed in Section D.1.5.2.

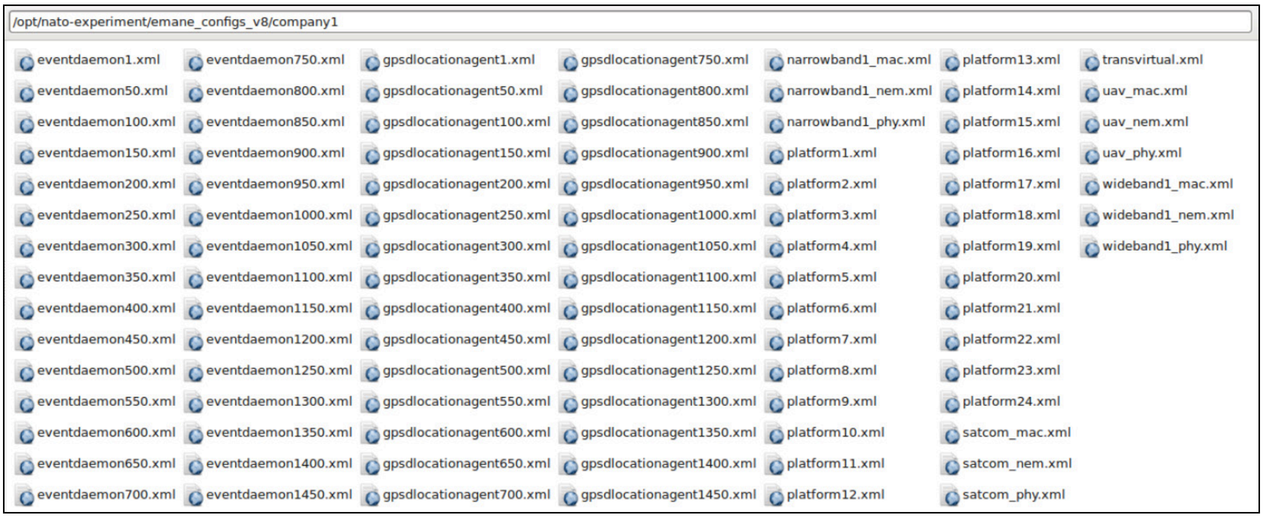

Figure D-28: Company1 EMANE Configuration Files.

D.1.5.3.1 Platform Configuration File

Figure D-29 shows the contents of ‘platform1.xml’, the platform XML file for Anglova scenario node company1-1. The top portion of the file specifies the configurations for the multicast channel EMANE uses to send and receive application traffic (Over-The-Air (OTA) group and device), and the multicast channel EMANE uses to receive location and path loss events (event service group and device). The values of the ‘otamanagerdevice’ and ‘eventservicedevice’ are set to the network interface associated with the DAVC network (172.15.0.0/23 – eth1) configured in Section D.1.4.3. The OTA service and the event service use the same device in this example, but they could be separated by setting either the ‘otamanagerdevice’ or ‘eventservicedevice’ to the second DAVC interface (172.16.0.0/23 - eth2) configured in Section D.1.4.3. The bottom portion contains 2 Network Emulation Module (NEM) entries for each of the radios this particular node is equipped with. A NEM is EMANE’s representation of an emulated radio. Note that the IP addresses defined for these NEMs are the same as what is defined in the network plan discussed in Section D.1.5.2.

D.1.5.3.2 Transport Configuration File

All of the nodes share the same ‘transvirtual.xml’ and radio model configuration files (wideband1, narrowband1, satcom, and uav). However, each radio within the ‘company1’ group has its own ‘eventdaemon.xml’ and ‘gpsdlocationagent.xml’ files. The naming conventions for these files are ‘eventdaemon<radio ID>.xml’ and ‘gpsdlocationagent<radio ID>.xml’, where the radio ID is one of the numbers in the green cells in the network plan file discussed in Section D.1.5.2. In EMANE terminology the radio ID is referred to as a NEM ID.

Figure D-30 shows the contents of the shared ‘transvirtual.xml’ file. This file is the same across all groups and therefore is shared. This file defines the transport library that provides the entry and exit point for the emulator and application space messages. The experimentation environment uses the Virtual Transport library, which uses a TAP interface to create a virtual interface as the application/emulation boundary entry/exit point. The virtual interfaces that will be created on a node are defined in the NEM/transport entries in the platform

XML file. Referring to the NEM/transport entries in Figure D-29, we can see that the DAVC VM node mapped to Anglova scenario node company1-1 node will have 2 virtual interfaces (‘emane0’ and ‘emane4’) created on it that will define the boundary between that node’s application space and the emulated radio.

```
<?xml version="1.0" ?><!DOCTYPE platform  SYSTEM 'file:///usr/share/emane/dtd/platform.dtd'>
<platform>
        <param name="otamanagerchannelenable" value="on"/>
        <param name="otamanagerdevice" value="eth1"/>
        <param name="otamanagergroup" value="224.1.2.8:45702"/>
        <param name="eventservicegroup" value="224.1.2.8:45703"/>
        <param name="eventservicedevice" value="eth1"/>
        <param name="controlportendpoint" value="0.0.0.0:47000"/>

        <nem definition="wideband1_nem.xml" id="1">
                <transport definition="transvirtual.xml">
                        <param name="device" value="emane0"/>
                        <param name="address" value="192.168.1.1"/>
                        <param name="mask" value="255.255.255.0"/>
                </transport>
        </nem>

        <nem definition="narrowband1_nem.xml" id="50">
                <transport definition="transvirtual.xml">
                        <param name="device" value="emane4"/>
                        <param name="address" value="192.168.5.1"/>
                        <param name="mask" value="255.255.255.0"/>
                </transport>
        </nem>
</platform>
```

OTA and Event Service Configurations

NEM 1 Radio Configurations

NEM 50 Radio Configurations

Figure D-29: Company1-1 Platform XML File.

```
<?xml version="1.0" ?>
<!DOCTYPE transport  SYSTEM 'file:///usr/share/emane/dtd/transport.dtd'>
<transport library="transvirtual" name="Virtual Transport">
        <param name="devicepath" value="/dev/net/tun"/>
</transport>
```

Figure D-30: Company1 Transvirtual XML File.

D.1.5.3.3 Event Daemon Configuration File

Figure D-31 shows the contents of ‘eventdaemon1.xml’, the event daemon settings for Anglova scenario node company1-1. In order for the scenario to progress, each NEM or radio must receive the location and path loss events for the each time step in the scenario. The event daemon listens to and receives events from the event channel. The event daemon’s (remaining) role is to make events available to ‘application space’ by means of its ‘agent’ plug-ins such as the gpsdlocationagent. The event daemon XML file defines the multicast group and interface where it will listen for events. Each NEM, as indicated by the ‘nemid’ value in this file, must have its own event daemon XML file defining these settings.

```
<?xml version="1.0" ?>
<!DOCTYPE eventdaemon  SYSTEM 'file:///usr/share/emane/dtd/eventdaemon.dtd'>
<eventdaemon nemid="1">
  <param name="eventservicegroup" value="224.1.2.8:45703"/>
  <param name="eventservicedevice" value="eth1"/>
  <agent definition="gpsdlocationagent1.xml"/>
</eventdaemon>
```

Figure D-31: Event Service Daemon XML File.

D.1.5.3.4 GPS Daemon Configuration File

This event daemon file also specifies specialized event agents that handle specific types of events. All of the nodes in the experimentation environment specify the GPSD location event agent shown in Figure D-32. This agent is responsible for making location events available as NMEA sentences, which can serve as input to user applications (e.g., GPSd, the GPS daemon) running on the node.

```
<?xml version="1.0" ?>
<!DOCTYPE eventagent  SYSTEM 'file:///usr/share/emane/dtd/eventagent.dtd'>
<eventagent library="gpsdlocationagent">
  <param name="pseudoterminalfile" value="/tmp/gps.pty"/>
</eventagent>
```

Figure D-32: GPSD Location Agent XML File.

D.1.5.3.5 Radio Model Configuration Files

The configuration files for the radio models represented within each group are located in the group directory also. Referring to Figure D-22 and Figure D-28, we see that company1 has nodes that will run the ‘wideband1’, ‘narrowband1’, ‘satcom’, and ‘uav’ radio models. Each of these radio model configuration files are located in the *‘/opt/nato-experiment/emane_configs_v8/company1’* directory and are shared amongst the nodes in the company1 group. A radio model is represented by 3 types of configuration files: a radio model NEM, MAC layer, and PHY layer file.

The ‘wideband1’ radio model NEM file is shown in Figure D-33. This file specifies the files that define the radio model’s MAC and PHY layer configurations. It also specifies the transport definition file previously discussed.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE nem SYSTEM 'file:///usr/share/emane/dtd/nem.dtd'>
<nem>
  <transport definition="transvirtual.xml"/>
  <mac definition="wideband1_mac.xml"/>
  <phy definition="wideband1_phy.xml"/>
</nem>
```

Figure D-33: Wideband1 Radio Model NEM XML File.

The ‘wideband1’ radio model PHY layer file is shown in Figure D-34. This file specifies the PHY layer library the radio will use (‘universalphy’) and contains the radio model’s physical layer properties such as bandwidth, frequency and transport model. It also sets the radio’s propagation model to precomputed, which means the NEM will be expecting to receive precomputed path loss events via the event service multicast channel previously discussed.

The ‘wideband1’ radio model MAC layer file is shown in Figure D-35. This file defines the MAC layer library (in this example: RFPipe) and contains the radio model’s MAC layer properties such as datarate, jitter and delay. It also defines the radio model’s packet completion rate curve XML file. This curve definition is comprised of a series of SINR (Signal to Interference plus Noise Ratio) values along with their corresponding probability of reception. The radio model uses the packet completion rate curve to determine an incoming packet’s probability of reception.

The experimentation environment consists of 17 different sets of radio model configuration files, all of which are similar in format to the configuration files discussed in this section but define different values for the various parameters.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE phy SYSTEM 'file:///usr/share/emane/dtd/phy.dtd'>
<phy name="universalphy">
  <param name="bandwidth" value="250K"/>
  <param name="frequency" value="300000000"/>
  <param name="frequencyofinterest" value="300000000"/>
  <param name="subid" value="1"/>
  <param name="systemnoisefigure" value="12.0"/>
  <param name="txpower" value="47.0"/>
  <param name="fixedantennagain" value="0.0"/>
  <param name="fixedantennagainenable" value="on"/>
  <param name="noisemode" value="none"/>
  <param name="noisebinsize" value="20"/>
  <param name="propagationmodel" value="precomputed"/>
</phy>
```

Figure D-34: Wideband1 Radio Model PHY Parameters.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mac SYSTEM 'file:///usr/share/emane/dtd/mac.dtd'>
<mac library="rfpipemaclayer" name="RF-PIPE MAC">
  <param name="enablepromiscuousmode" value="off"/>
  <param name="datarate" value="175K"/>
  <param name="flowcontrolenable" value="off"/>
  <param name="flowcontroltokens" value="10"/>
  <param name="pcrcurveuri" value="/usr/share/emane/xml/models/mac/rfpipe/rfpipepcr.xml"/>
  <param name="jitter" value="0.0"/>
  <param name="delay" value="0.0"/>
</mac>
```

Figure D-35: Wideband1 Radio Model MAC Parameters.

D.1.5.4 EMANE Event Service Configuration Files

The EMANE event service is responsible for generating mobility and pathloss events for the NEMs running within the emulation. The EMANE event service takes as its input several XML configuration files as well as event files in Emulation Event Log (EEL) format to function correctly. The EMANE mobility and pathloss event configuration files for the Anglova scenario vignettes are located in the */opt/nato-experiment/event_service_configs* directory shown in Figure D-36.

D.1.5.4.1 Event Service XML

The event service XML configuration file (Figure D-37) defines the multicast channel and interface where events will be published. It also defines the EEL generator configuration file, which contains event parser and multicast channel configurations.

D.1.5.4.2 EEL Generator XML

The EEL generator configuration file (Figure D-38) defines the source EEL event file and the various parser plugins that will be used to parse sentences from the EEL event file.

D.1.5.4.3 EEL Source File

Mobility and pathloss events are stored in and parsed from emulation event log files (Figure D-39). These files consist of time stamped entries for each NEM's GPS location (latitude, longitude, altitude) as well as time stamped entries for the pathloss values between the nodes. Mobility and pathloss events may be combined into a single file or separated into individual files.

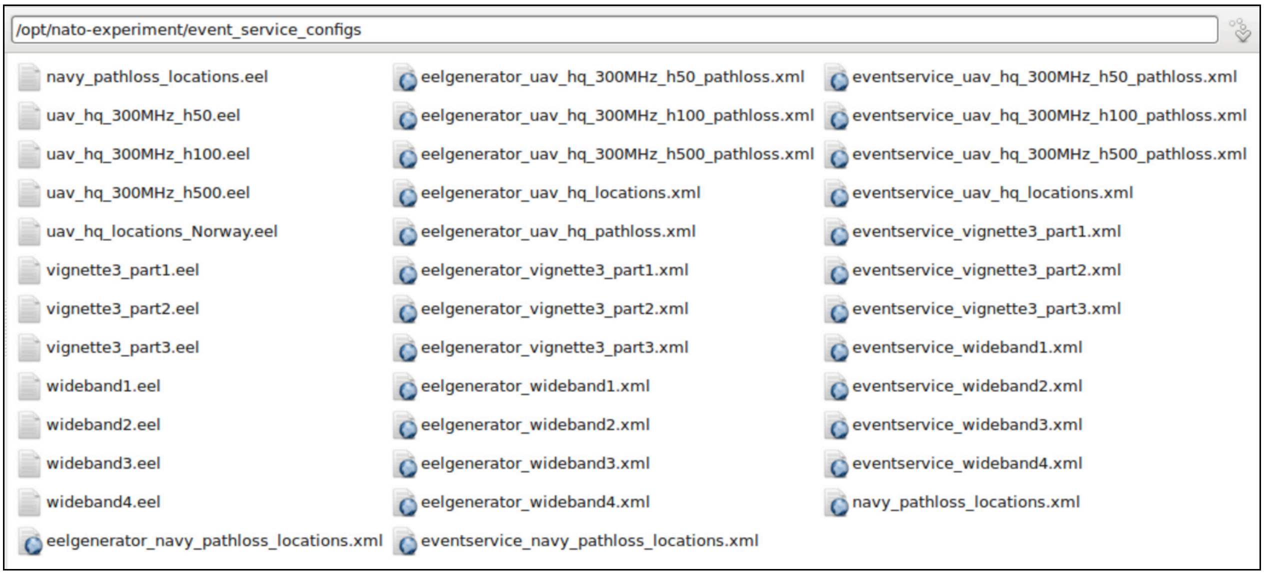

Figure D-36: EMANE Event Service Configuration File System.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE eventservice SYSTEM "file:///usr/share/emane/dtd/eventservice.dtd">
<eventservice>
  <param name="eventservicegroup" value="224.1.2.8:45703"/>
  <param name="eventservicedevice" value="eth1"/>
  <generator definition="eelgenerator_vignette3_part1.xml"/>
</eventservice>
```

Figure D-37: Example Event Service Configuration File.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE eventgenerator SYSTEM "file:///usr/share/emane/dtd/eventgenerator.dtd">
<eventgenerator library="eelgenerator">
    <param name="inputfile" value="vignette3_part1.eel" />
    <paramlist name="loader">
      <item value="commeffect:eelloadercommeffect:delta"/>
      <item value="location,velocity,orientation:eelloaderlocation:delta"/>
      <item value="pathloss:eelloaderpathloss:delta"/>
      <item value="antennaprofile:eelloaderantennaprofile:delta"/>
    </paramlist>
</eventgenerator>
```

Figure D-38: Example EEL Generator Configuration File.

```
0.0 nem:12350 location gps 58.170309,7.944924,2.0
0.0 nem:12350 pathloss nem:12550,0.0 nem:12650,0.0 nem:12700,0.0 nem:12750,0.0 nem:12800,0.0 nem:12850,0.0 nem:12900,0.0 nem:12950,0.0 nem:13000,0.0 nem:1305
0,0.0 nem:13100,26.2 nem:13150,26.2 nem:13250,26.2 nem:13300,26.2 nem:13350,26.2 nem:13400,26.2 nem:13450,26.2 nem:13500,26.2 nem:13550,26.2 nem:13600,26.2 n
em:13650,26.2 nem:13700,0.0 nem:13760,26.2
0.0 nem:12400 location gps 58.170309,7.944924,2.0
0.0 nem:12400 pathloss nem:12600,0.0 nem:13200,10.7 nem:13850,17.3 nem:14100,17.3 nem:14700,17.3 nem:15350,19.5 nem:15550,19.5 nem:16150,10.7
0.0 nem:12450 location gps 58.170309,7.944924,2.0
0.0 nem:12450 pathloss nem:13900,32.9 nem:15400,35.0
0.0 nem:12500 location gps 58.170309,7.944924,2.0
0.0 nem:12500 pathloss nem:13740,0.0 nem:13775,10.7 nem:14000,17.3 nem:15180,17.3 nem:15250,17.3 nem:15450,19.5 nem:16680,19.5 nem:16750,10.7 nem:16800,55.9
0.0 nem:12550 location gps 58.170309,7.944924,2.0
0.0 nem:12550 pathloss nem:12350,0.0 nem:12650,0.0 nem:12700,0.0 nem:12750,0.0 nem:12800,0.0 nem:12850,0.0 nem:12900,0.0 nem:12950,0.0 nem:13000,0.0 nem:1305
0,0.0 nem:13100,26.2 nem:13150,26.2 nem:13250,26.2 nem:13300,26.2 nem:13350,26.2 nem:13400,26.2 nem:13450,26.2 nem:13500,26.2 nem:13550,26.2 nem:13600,26.2 n
em:13650,26.2 nem:13700,0.0 nem:13760,26.2
```

Figure D-39: Example EEL File.

D.1.5.5 Experimentation Scripts

There are two scripts that manage the starting and stopping of the experimentation environment, ***start_anglova_experiment.sh*** and ***stop_anglova_experiment.sh***. These two scripts are the only scripts that are explicitly executed by the user on the command line. The other scripts that will be discussed (***start_emane.sh**** and ***stop_emane.sh***) are not explicitly executed by the user, but are instead executed indirectly by the ***start_anglova_experiment.sh*** and ***stop_anglova_experiment.sh*** scripts.

D.1.5.5.1 start_anglova_experiment.sh

The ***start_anglova_experiment.sh*** script is responsible for starting the EMANE emulation of the various Anglova vignettes. The ***start_anglova_experiment.sh*** input parameters are shown in Figure D-40. This script requires 3 parameters:

- (-c) The name of the DAVC experimentation cluster that was created in Section D.1.4.2.
- (-n) The number of VMs that are available in the cluster to be mapped to Anglova scenario nodes. This number should not include the experimentation controller VM.
- (-v) The Anglova vignette that will be executed. The possible options are:
  - 2: Vignette 2 (Deployment of the Coalition Forces).
    - 3-1: Vignette 3 Part 1 (Insurgent Neutralization).
    - 3-2: Vignette 3 Part 2 (IED Neutralization).
    - 3-3: Vignette 3 Part 3 (Medevac of Wounded).
  - Custom: Custom vignette where the user specifies which groups will be executed.
  - (-s) The IP Address of the SDT-3D visualization application (see Section D.1.6.1).

```
start_anglova_experiment.sh [-h] [-c Cluster Name] [-n Available VMs] [-v Vignette] [-s SDT-3D IP]
 -- Starts the EMANE Emulation of the specified Anglova Vignette

where:
    -h  Shows this help text.
    -c  Experimentation cluster name.
    -n  Number of available virtual machines excluding the experiment controller.
    -v  Anglova Vignette to start:
                2       = Vignette 2
                3-1     = Vignette 3 part 1
                3-2     = Vignette 3 part 2
                3-3     = Vignette 3 part 3
                custom  = Custom Vignette (requires editing this script)
    -s  IP Address of the SDT-3D Visualization Application
```

Figure D-40: start_anglova_experiment.sh Script Usage.

When executed the script file does the following:

1) Determines which groups in the Anglova scenario should be started in the experimentation environment based on the specified vignette parameter.
2) Cycles through the DAVC VM nodes sequentially and assigns each to a group node from the scenario. For example, when executing Vignette 2, the first group node from company1, company1-1, will be assigned to and started on the first DAVC VM node exp-1, company1-2 on exp-2 and so on.
3) Reads the network plan file to determine which radio models to start on the chosen node.
4) Remotely launches the ***start_emane.sh*** script with the corresponding EMANE radio model configuration files on the chosen node. The ***start_emane.sh*** will be discussed next but its execution will ultimately result in the creation of the virtual EMANE interfaces on the chosen node as discussed in Section D.1.5.3.
5) Creates an updated host file with host names for each Anglova scenario node in the experiment corresponding to the radios they possess. This host file is copied to each of the DAVC VM nodes. The host file is based on the radio host name and IP addresses outlined in the network plan as shown in Section D.1.5.2, Figure D-24 and Figure D-25.
6) Starts the EMANE event service to begin sending the specified vignette's location and path loss events to EMANE event service multicast channel.

*D.1.5.5.2 start_emane.sh**

The ***start_emane.sh**** scripts are not executed directly by the user but are executed indirectly and remotely by the ***start_anglova_experiment.sh*** script as discussed in the previous section. Shown in Table D-4 are the 3 versions of the ***start_emane.sh**** script that starts the EMANE emulator components on the DAVC VM node, however the ***start_emane_olsrv1.sh*** and the ***start_emane_olsrv2.sh*** scripts also start the OLSRv1 or OLSRv2 routing protocols respectively. The ***start_emane_none.sh*** script does not start a routing protocol. Note that ***start_emane.sh*** is simply a symbolic link that should be set to point to one of the other *start_emane.sh** scripts depending on if a routing protocol should be run or not.

Table D-4: The Different Version of the Script that Starts EMANE.

File	Description
start_emane.sh	Symbolic link that points to one of the other start_emane.sh_<routing protocol version> scripts.
start_emane_none.sh	Script to start the EMANE emulator without a routing protocol on a DAVC cluster node.
start_emane_olsrv1.sh	Script to start the EMANE emulator with the OLSRv1 routing protocol on a DAVC cluster node.
start_emane_olsrv2.sh	Script to start the EMANE emulator with the OLSRv2 routing protocol on a DAVC cluster node.

When executed, the ***start_emane.sh**** script files do the following:

1) Starts the EMANE executable with the EMANE platform configuration file (see Figure D-29) for the assigned Anglova scenario node. This results in the creation of the virtual EMANE interfaces specified in that node's *platform.xml* file.

2) Starts the EMANE event daemon executable with the EMANE event daemon configuration file (see Figure D-31) for each NEM present on the Anglova scenario node.

3) Starts the GPS Daemon executable. The ***gpsd*** service collects information from a specified GPS source.

4) If ***start_emane.sh_olsrv1*** or ***start_emane.sh_olsrv2*** is selected, launches the Optimized Link State Routing (OLSR) protocol on the EMANE interfaces on the chosen node (Figure D-19). For example, on node company1-1, OLSRv1 or OLSRv2 will be started on its emane0 and emane4 network interfaces.

D.1.6 Launching an Anglova Vignette

Now that the experimentation environment's DAVC cluster has been configured and the environment's experimentation scripting has been reviewed, an Anglova scenario emulation can be run. In this section the instructions to launch Anglova Vignette 2 using the DAVC experimentation cluster configured in Section D.1.4 will be outlined. Vignette 2 covers the deployment of the coalition forces, a battalion consisting of 157 nodes, into the operational zone as discussed in Section D.1.2. The same instructions can be used to launch Anglova Vignette 3 parts 1, 2, and 3 with the only difference being the input parameters provided to the ***start_anglova_experiment.sh*** experimentation scripting discussed in the previous section.

D.1.6.1 Configure Scenario Visualization

The IST-124-061 experimentation environment uses the Naval Research Lab's Scripted Display Tools (SDT-3D) [5] to visualize the emulated scenario's nodes, mobility, links and connectivity on a NASA Whirlwind geographic background. See Figure D-41.

Figure D-41: SDT-3D Visualization Tool.

The experimentation environment template VM contains an EMANE SDT-3D client/server framework (Figure D-42) that enables the sending of visualization commands to a running instance of SDT-3D to visualize the running EMANE emulation. When the experimentation environment is started, the EMANE SDT-3D client starts automatically on each scenario node's VM. This client reads location and network connectivity from the EMANE software and sends that information to the EMANE SDT-3D server, which is started on the experimentation controller VM. The EMANE SDT-3D server uses this information to generate and send SDT-3D commands to visualize the running emulation. The receiving SDT-3D application can run on an external machine as long as that machine has network connectivity to the experimentation controller.

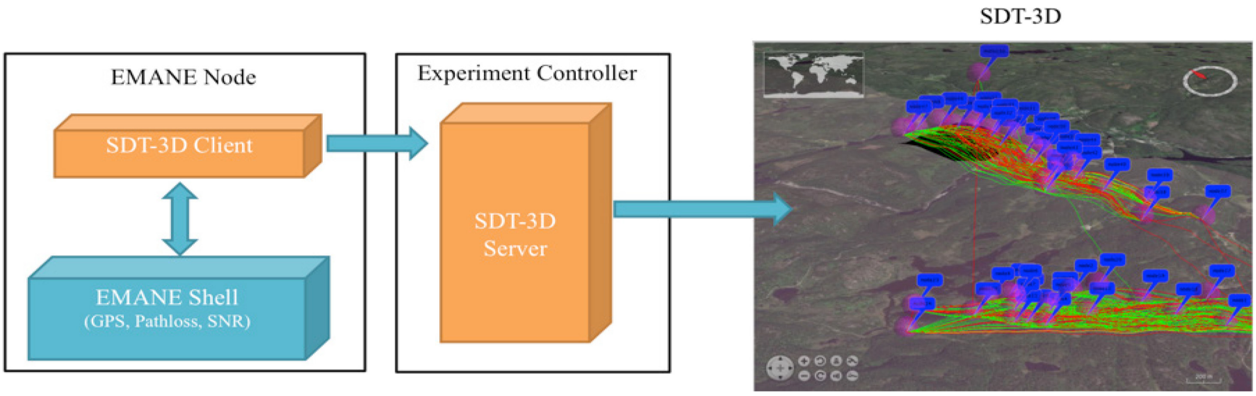

Figure D-42: EMANE SDT-3D Visualization Client/Server Framework.

To configure the SDT-3D application to listen and receive scenario visualization commands from the EMANE SDT-3D server, open the SDT-3D application and in the *'**File**'* menu select the *'**Listen to TCP port…**'* option (Figure D-43). Enter port '*__55002__*' into the input dialog that appears and press '*__OK__*'. SDT-3D is now configured to listen for visualization commands from the EMANE SDT-3D server. The EMANE SDT-3D server will be configured to send the visualization commands to the SDT-3D application in a later step.

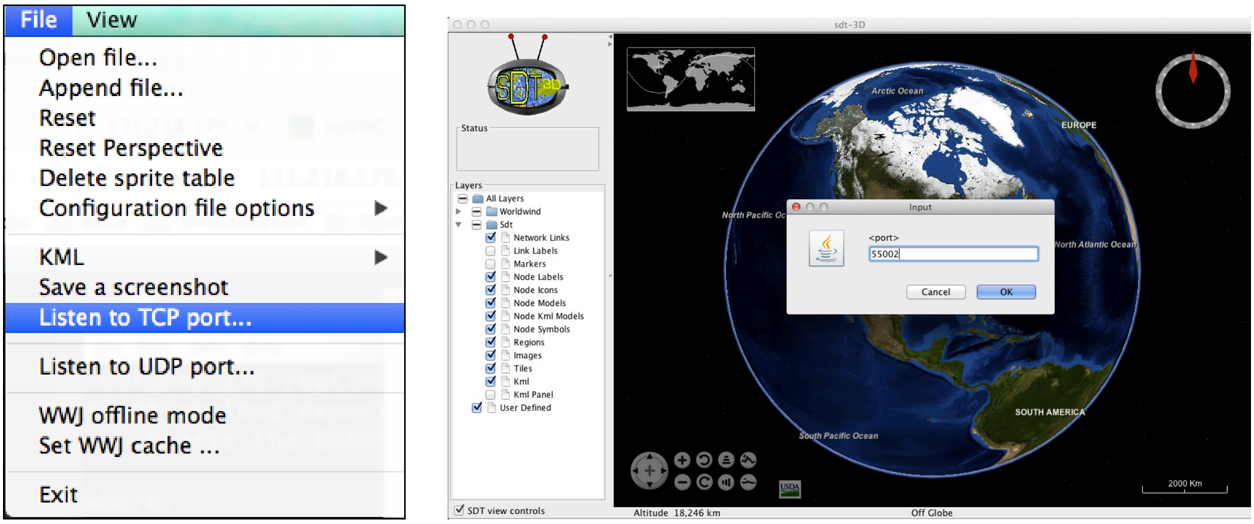

Figure D-43: Configure SDT-3D to Listen on TCP Port 55002.

The EMANE SDT-3D framework will generate an SDT-3D log file containing all of the events processed during the scenario. This file can be loaded into the SDT-3D application to replay the scenario visualization. This feature is useful if an EMANE SDT-3D server is not available. The log file will be located on the experimentation controller in *'/log/<timestamp>.sdt'* where *'<timestamp>'* is the time the experimentation scenario was run. To configure the SDT-3D application to load and replay the scenario visualization, open the SDT-3D application and in the *'File'* menu select the *'Open file…'* option (Figure D-44). Navigate to the SDT-3D log file and press '*OK*'.

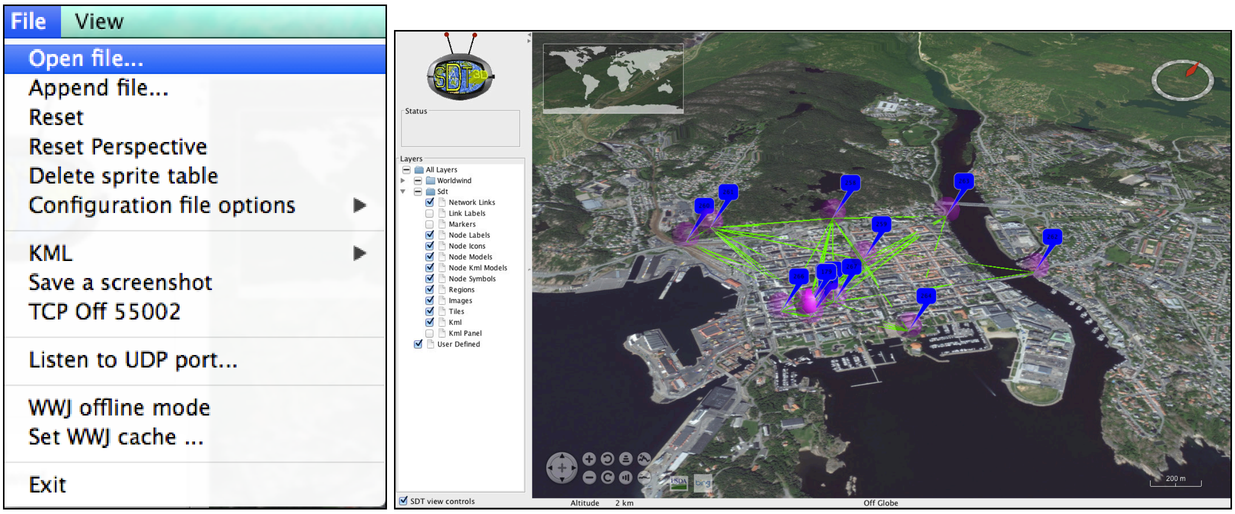

Figure D-44: Configure SDT-3D to Load and Replay an SDT-3D Log File.

D.1.6.2 Log into the Experimentation Controller

The vignette will be executed from the experimentation controller node. From the experiment's DAVC details page, log into VM node 270's Virtual Network Computing (VNC) console by clicking on its *'Open VNC'* button in the *'Node Options'* dropdown menu (Figure D-45).

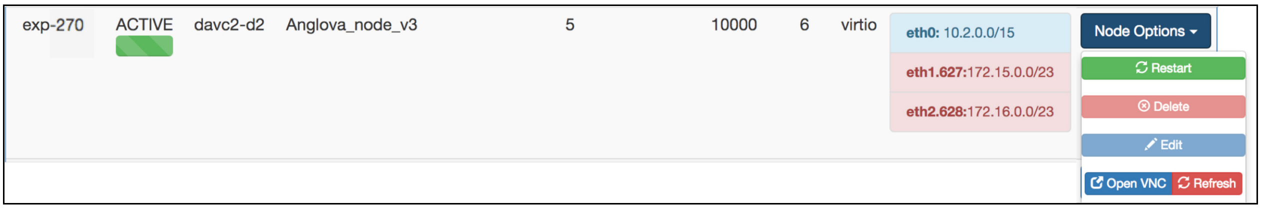

Figure D-45: Log into VM Node 270's VNC Console.

Next, open a console terminal and navigate to the experimentation script's home directory */opt/nato-experiment* (Figure D-46).

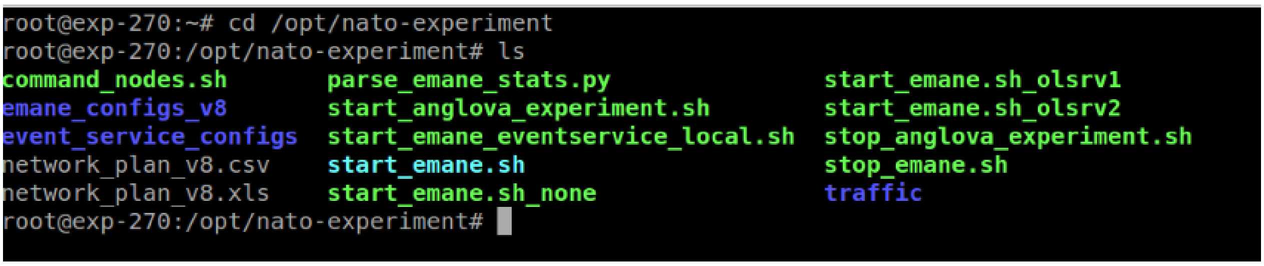

Figure D-46: Navigate to the Experimentation Scripts Home Directory on Node Exp-270.

Next, to start Vignette 2, execute the ***start_anglova_experiment.sh*** script as shown in Figure D-47, using the following command: ***./start_anglova_experiment.sh -c exp -n 269 -v 2 –c 10.2.1.40***. The parameters used in this command specify that Vignette 2 (***-v 2***) will be executed in the 'exp' DAVC cluster (***-c exp***), which has 269 available cluster nodes (***-n 269***). The command also specifies the SDT-3D application's IP Address (***-s 10.2.1.40***) where the EMANE SDT-3D server discussed in Section D.1.6.1 will send visualization commands. The script will launch the experimentation components (EMANE, EMANE event daemon, GPS daemon, etc.) on each scenario node within the DAVC cluster nodes.

```
root@exp-270:/opt/nato-experiment# ./start_anglova_experiment.sh -c exp -n 269 -v 2 -s 10.2.1.40
Adding multicast route, 224.1.2.8, to eth1
Refreshing SDT-3D Visualization Server...
Stopping emanesdtfrmwrk_sdt_set_manager_1 ... done
Stopping emanesdtfrmwrk_emane_data_queue_1 ... done
Stopping emanesdtfrmwrk_redis_server2_1 ... done
Stopping emanesdtfrmwrk_redis_server_1 ... done
Removing emanesdtfrmwrk_sdt3d_driver_1 ... done
Removing emanesdtfrmwrk_sdt_set_manager_1 ... done
Removing emanesdtfrmwrk_emane_data_queue_1 ... done
Removing emanesdtfrmwrk_redis_server2_1 ... done
Removing emanesdtfrmwrk_redis_server_1 ... done
Creating emanesdtfrmwrk_redis_server_1
Creating emanesdtfrmwrk_redis_server2_1
Creating emanesdtfrmwrk_emane_data_queue_1
Creating emanesdtfrmwrk_sdt_set_manager_1
Creating emanesdtfrmwrk_sdt3d_driver_1

Starting company1 platforms: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24

Starting scenario node company1-1
Scenario node company1-1 assigned to DAVC cluster node exp-1
NEMS allocated to company1-1: 1 50

Copying script files to exp-1...

start_emane.sh                                                                    100% 1548     1.5KB/s   00:00
stop_emane.sh                                                                     100%  387     0.4KB/s   00:00
hosts                                                                             100%   23KB  23.0KB/s   00:00

Starting EMANE on node exp-1...

Clearing experiment logs...
rm: cannot remove '/log/*.pcap': No such file or directory
rm: cannot remove '/log/*.tar.gz': No such file or directory
Adding multicast route, 224.1.2.8, on eth1...
SIOCADDRT: File exists
Starting EMANE....
emane /opt/nato-experiment//emane_configs_v8/company1/platform1.xml -r -d -l 4 -f /log/emane1.log
Starting EMANE event daemon for NEM 1....
```

Figure D-47: Vignette 2 Execution Output.

The script also performs other actions including launching an EMANE SDT-3D visualization framework client and OLSR routing (Figure D-48) on the cluster nodes as well. The EMANE event service discussed in Section D.1.5.4 is also started on the experimentation controller node with the corresponding mobility and pathloss EEL files for Vignette 2.

After the ***start_anglova_experiment.sh*** script completes the vignette is now running. The SDT-3D instance will begin to update showing the vignette's nodes, their mobility and connectivity (Figure D-49). Refer to the network plan discussed in D.1.5.2 to identify the nodes involved in the vignettes. Each node can be accessed via their DAVC VNC console or via ssh using their DAVC node names as host names. The network plan also contains the host names and IP addresses for the emulated EMANE radios.

The instructions outlined in this section can be used to run Vignette 3, just specify the appropriate vignette parameter to the ***start_anglova_experiment.sh***.

D.1.7 Launching a CUSTOM Anglova Vignette

The ***start_anglova_experiment.sh*** script can also be used to launch custom vignettes. When running a custom vignette the user specifies which groups should be activated in the vignette. This allows users to run subsets of the nodes in a particular vignette. The group-to-vignette mapping spreadsheet (Figure D-26) discussed in Section D.1.5.2 is especially useful in determining viable custom vignette group combinations. For example, if a user is only interested in running Vignette 3 part 2 with Platoon1 and the Unattended Ground Sensor (UGS) nodes, the user can achieve this by using the '***-v custom***' command line parameter.

In addition, the user would not be required to instantiate a DAVC cluster with all 270 VMs. Instead, a DAVC cluster with 35 VMs would be sufficient to run a custom vignette that includes Platoon1 (24 VMs/nodes) and the UGS (10 VMs/nodes). The last VM would be used as the experimentation controller.

```
Starting OLSR Routing....

 *** olsr.org -  0.6.6.2-git_0000000-hash_12fa41b5362519d37bea6715ce291978 ***
 Build date: 2014-09-11 13:09:19 on toyol
 http://www.olsr.org

Parsing file: "/opt/nato-experiment//emane_configs_v8/routing.conf"
Debug level: 0
IpVersion: 4
Lock file /log/olsrd.lock
Clear screen enabled
Link quality aging factor 0.200000
HNA IPv4 entry: 172.100.19.3/255.255.255.255
setting ifs_in_curr_cfg = 0
        HELLO interval: 2.00
        HELLO validity: 20.00
        TC interval: 8.00
        TC validity: 80.00
        IPv4 broadcast/multicast : AUTO
        Mode             : mesh
        IPv6 multicast           : ::
        HELLO emission/validity  : 2.00 (d)/20.00 (d)
        TC emission/validity     : 8.00 (d)/80.00 (d)
        MID emission/validity    : 0.00/0.00
        HNA emission/validity    : 0.00/0.00
        Autodetect changes       : no
        IPv4 broadcast/multicast : AUTO
        Mode             : mesh
        IPv6 multicast           : ::
        HELLO emission/validity  : 2.00 (d)/20.00 (d)
        TC emission/validity     : 8.00 (d)/80.00 (d)
        MID emission/validity    : 0.00/0.00
        HNA emission/validity    : 0.00/0.00
        Autodetect changes       : no
        IPv4 broadcast/multicast : AUTO
        Mode             : mesh
        IPv6 multicast           : ::
```

Figure D-48: Vignette 2 Execution Output.

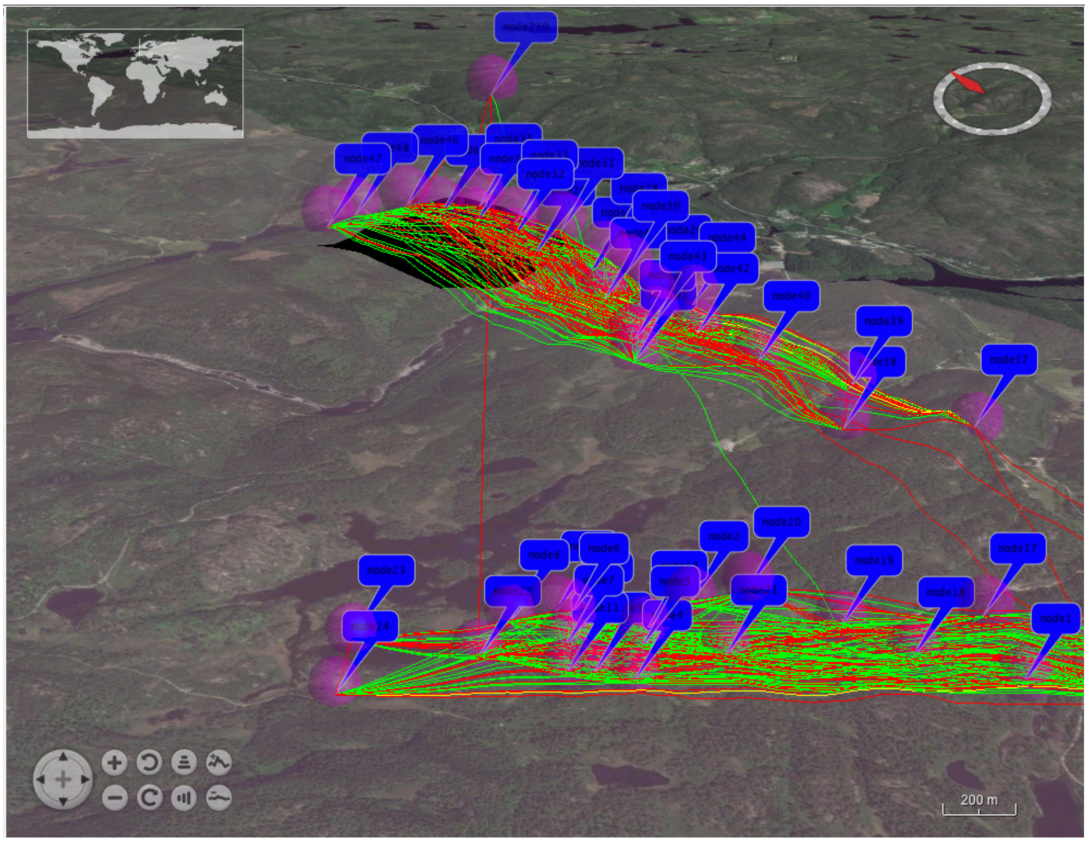

Figure D-49: Vignette 2 Emulation Visualization.

To configure this custom vignette create a DAVC cluster following the steps outlined in Section D.1.4, but instead of configuring 269 VMs, configure 34 VMs as shown in Figure D-50. These VMs will host the 24 Platoon1 and 10 UGS nodes. In general, to calculate the correct number of DAVC VM nodes to configure when preparing a custom vignette, refer to the network plan files discussed Section D.1.5.2. However, it is not a problem to have more VMs in a cluster than are actually assigned to emulated network nodes unless DAVC resources are scarce. In some experimentation setups, having more nodes may be more efficient than creating and deleting DAVC clusters for each individual experiment. Creating a single cluster to accommodate the largest custom vignette would allow different sized vignette experiments to be run in quick succession using the same cluster.

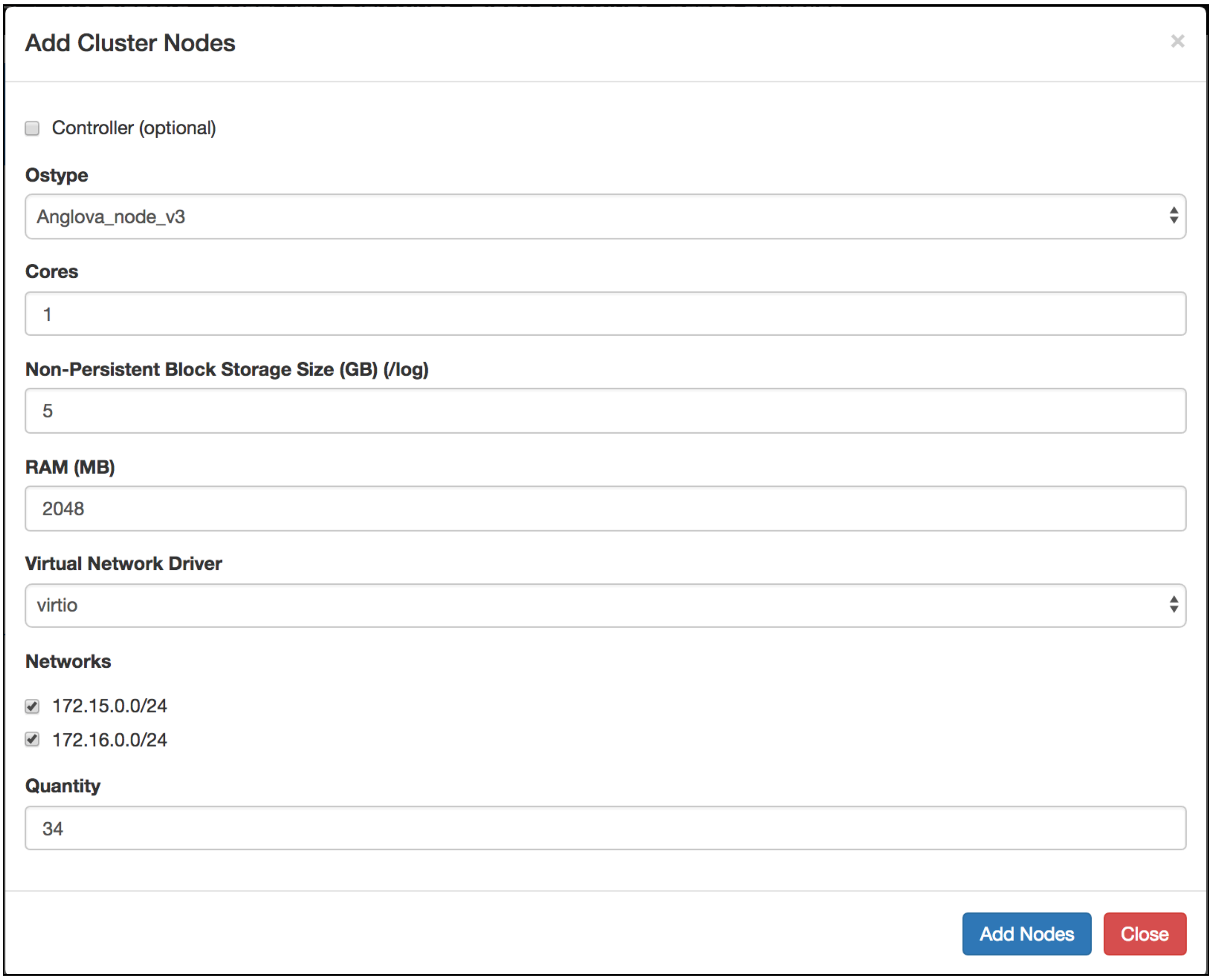

Figure D-50: Example Custom Vignette DAVC Configuration (Platform Nodes).

Configure an additional DAVC VM for the experimentation controller (Figure D-51). The VM should be configured just like the experimentation controller VM outlined in Section D.1.4. The steps to launch and activate the DAVC cluster are the same as in Section D.1.4 with the exception that the experimentation controller is now VM node ***exp-35*** instead of ***exp-270.***

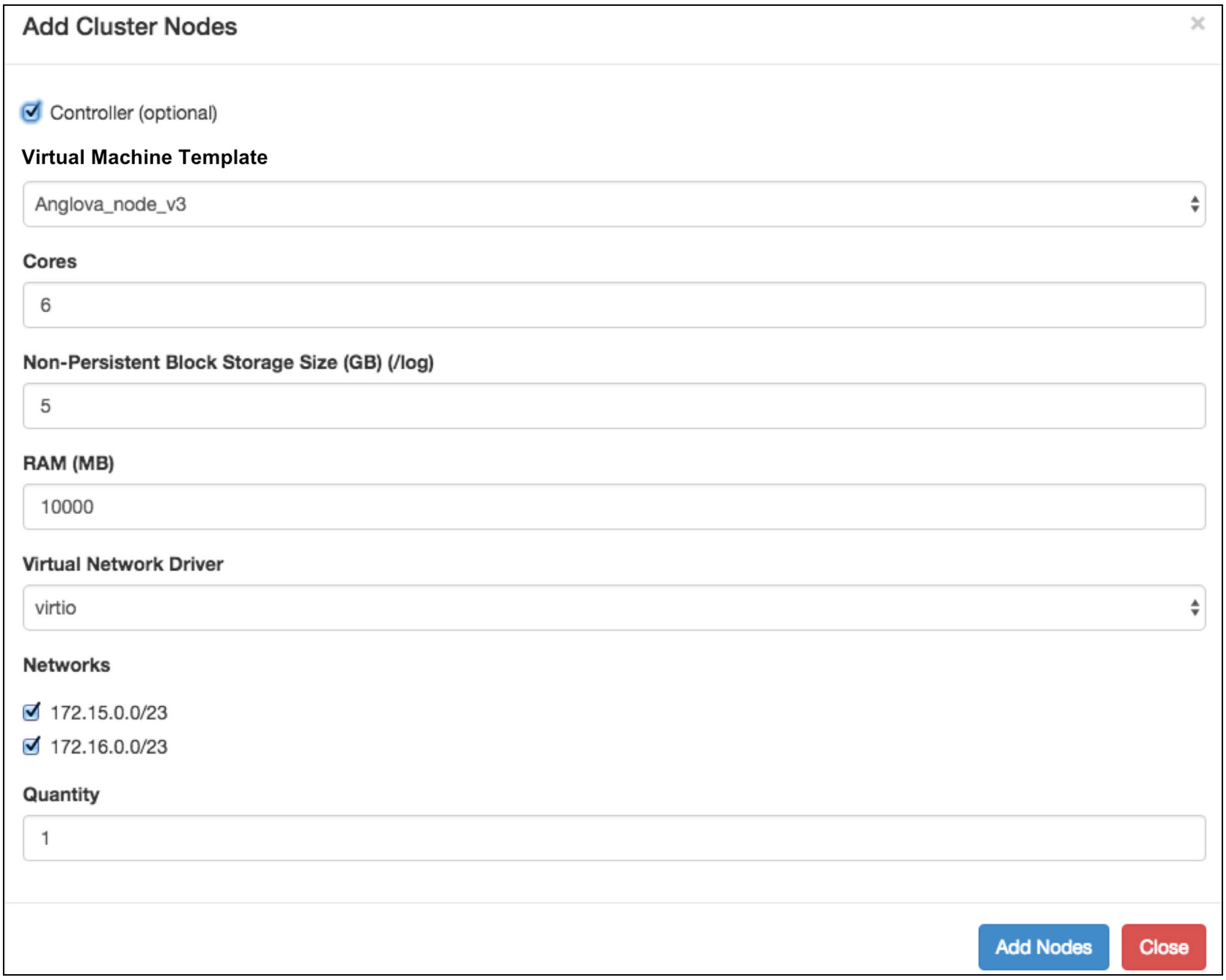

Figure D-51: Example Custom Vignette DAVC Configuration (Controller Node).

The steps to configure the SDT-3D application are the same as outlined in Section D.1.6.1, so no changes are required.

Next, log into the experimentation controller DAVC VNC console where the ***start_anglova_experiment.sh*** script will be run. However, when running a custom vignette the user must first edit the script to specify the groups that will be included in the vignette.

Navigate to the ***/opt/nato-experiment*** directory and open the ***start_anglova_experiment.sh*** script file with an editor. Edit the ***"ACTIVE_GROUPS"*** variable definition starting on line 208 by marking ***'true'*** for each group that should be included and ***'false'*** for the groups that should not be included. Save and close the file. Figure D-52 shows the ***"ACTIVE_GROUPS"*** definition for the custom vignette with Platoon1 and the UGS. The ***'platoon1*'*** and ***'ugs*'*** groups are marked ***'true'*** while all other groups are marked ***'false'***.

Next, to start the custom vignette, execute the ***start_anglova_experiment.sh*** script as shown in Figure D-53, using the following command: ***./start_anglova_experiment.sh -c exp -n 34 -v custom –c 10.2.1.40***. The parameters used in this command specify that a custom vignette will be launched (***-v custom***) in the 'exp' DAVC cluster (***-c exp***), which has 34 available cluster nodes (***-n 34***). The command also specify the SDT-3D application's IP Address (***-s 10.2.1.40***) where the EMANE SDT-3D server discussed in Section D.1.6.1 will send visualization commands.

```
    "custom")

            declare -A ACTIVE_GROUPS=(
            [company1]="false"
            [company2]="false"
            [company3]="false"
            [company4]="false"
            [command]="false"
            [support1]="false"
            [support2]="false"
            [geo-sat]="false"
            [harvest-uav]="false"
            [iridium-sat]="false"
            [med-helicopter]="false"
            [navy]="false"
            [platoon1]="true"
            [platoon1-APC]="true"
            [platoon1-commander]="true"
            [platoon1-section-leader]="true"
            [platoon2]="false"
            [platoon2-APC]="false"
            [platoon2-commander]="false"
            [platoon2-section-leader]="false"
            [platoon3]="false"
            [platoon3-APC]="false"
            [platoon3-commander]="false"
            [platoon3-section-leader]="false"
            [tac-sat]="false"
            [tac-uav]="false"
            [toc-hq]="false"
            [ugs]="true"
            [ugs-gw]="true"
            [ugs-relay]="true"
            )
            ;;
```

Figure D-52: Edit the start_anglova_experiment.sh Script for Custom Vignettes.

```
root@exp-270:/opt/nato-experiment# ./start_anglova_experiment.sh -c exp -n 34 -v custom -s 10.2.1.40
Adding multicast route, 224.1.2.8, to eth1
Refreshing SDT-3D Visualization Server...
Stopping emanesdtfrmwrk_sdt3d_driver_1 ... done
Stopping emanesdtfrmwrk_emane_data_queue_1 ... done
Stopping emanesdtfrmwrk_redis_server_1 ... done
Stopping emanesdtfrmwrk_redis_server2_1 ... done
Removing emanesdtfrmwrk_sdt3d_driver_1 ... done
Removing emanesdtfrmwrk_sdt_set_manager_1 ... done
Removing emanesdtfrmwrk_emane_data_queue_1 ... done
Removing emanesdtfrmwrk_redis_server_1 ... done
Removing emanesdtfrmwrk_redis_server2_1 ... done
Creating emanesdtfrmwrk_redis_server2_1
Creating emanesdtfrmwrk_redis_server_1
Creating emanesdtfrmwrk_emane_data_queue_1
Creating emanesdtfrmwrk_sdt_set_manager_1
Creating emanesdtfrmwrk_sdt3d_driver_1

Starting platoon1 platforms: 160 161 162 163 164 165 166 167 168 169 171 172 173 174 175 176 177 178 179

Starting scenario node platoon1-1
Scenario node platoon1-1 assigned to DAVC cluster node exp-1
NEMS allocated to platoon1-1: 12650

Copying script files to exp-1...

start_emane.sh                                                                                                                100% 1548     1.5KB/s   00:00
stop_emane.sh                                                                                                                 100%  387     0.4KB/s   00:00
hosts                                                                                                                         100%   23KB  23.0KB/s   00:00

Starting EMANE on node exp-1...

Clearing experiment logs...
rm: cannot remove '/log/*.pcap': No such file or directory
rm: cannot remove '/log/*.tar.gz': No such file or directory
Adding multicast route, 224.1.2.8, on eth1...
SIOCADDRT: File exists
Starting EMANE....
emane /opt/nato-experiment//emane_configs_v8/platoon1/platform160.xml -r -d -l 4 -f /log/emane160.log
Starting EMANE event daemon for NEM 12650....
```

Figure D-53: Example Custom Vignette Execution.

After the script has completed, the custom vignette emulation will be running on the cluster nodes. The SDT-3D instance will begin to update showing the specified vignette nodes, their mobility and connectivity. Each node can be accessed via their DAVC VNC console or via ssh using their DAVC node names as host names. The network plan also contains the host names and IP addresses for the emulated EMANE radios.

D.1.8 Conclusion

The NATO-IST-124 experimentation environment provides a common platform to explore research issues relevant to heterogeneous tactical networks, including routing topology architectures and their impact on delivery rates, overheads, and scalability; data dissemination protocols; quality of service and resource management; and leveraging and integration of sensor networks. This portion of the annex detailed the steps required to launch the EMANE emulation of the IST-124 Anglova experimentation scenario within ARL's DAVC environment. The instructions provided can be used as a guide to launch various subsets of the entire 269-node emulation scenario for a wide range of experimentation backdrops.

D.2 DYNAMICALLY ALLOCATED VIRTUAL CLUSTER (DAVC) MANAGEMENT SYSTEM V2.0 SETUP GUIDE

The Dynamically Allocated Virtual Clustering Management System (DAVC) is an experimentation infrastructure that provides the means to dynamically create, deploy, and manage virtual clusters of heterogeneous nodes within a cloud computing environment. The system allows researchers to create clusters that can be utilized for software development, experimentation, and integration with existing hardware and software. DAVC is built on proven technologies that are open, scalable, and well documented. The system can deploy both stateless nodes via network booting and nodes from Virtual Hard Drives containing a preinstalled operating system. It uses Kernel-based Virtual Machines (KVM) and Quick EMUlator (QEMU), a full virtualization solution where each virtual machine has private virtual hardware. It also interfaces with Oracle Grid Engine Distributed Resource Management System (DRM) to dynamically assign Virtual Machines (VMs) to hardware resources based on CPU, RAM, hard disk and network traffic. This document is a guide for DAVC system setup and configuration. Please refer to the DAVC User Guide in Section D.3 for specific DAVC usage details.

D.2.1 System Layout

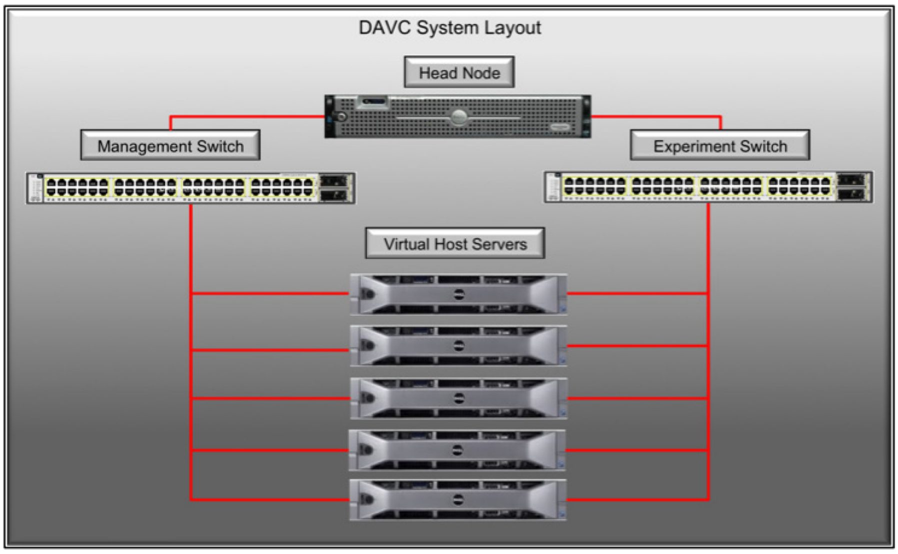

Figure D-54: DAVC System Layout.

Notes:

The characters for single (’) and double quotes (“) do not always translate correctly if copying from the document to a terminal window. Be prepared to retype quotes if problems arise.

Every command is run as root unless otherwise stated.

D.2.2 Assumptions

This document makes the following assumptions.

D.2.2.1 Number of Systems

The Dynamically Allocated Virtual Cluster (DAVC) Management System will consist of a DAVC Controller (DAVC) and at least two Virtual Host Servers (d1-d2). You may add as many Virtual Host Servers as needed.

D.2.2.2 Operating System

The system will use Ubuntu 14.04 Server 64-bit on both the DAVC Controller and the Virtual Host Servers. Other recent versions of Ubuntu or Debian could be used if one is willing to experiment.

D.2.2.3 Network

The system will consist of two main physical networks: Management and Private (referred to by the name “experiment”). By default, neither of these networks resides on the public Internet. The administrator of the system must have full control over the Management and Private networks. This includes configuring the physical switches and the IP space. In addition, the DAVC Controller contains a link to the Internet, which is called Public. If desired, Network Address Translation (NAT) routing can be configured between the Public and the Management networks. The IP spaces used throughout this installation guide are examples and should be replaced accordingly by the administrator of the system.

D.2.2.3.1 Virtual Host Servers Network Ports

Note:

The number of ports for the Management and Private bridges can be different across host servers.

- 1 x 1 Gb Ethernet Port for Base Management IP address;
- 4 x 1 Gb Ethernet Port for Management network bridges; and
- 4 x 1 Gb Ethernet Port for Private network bridges.

D.2.2.3.2 Switches

- 1 x Cisco 3750-E Switch (Management); and
- 1 x Cisco 3750-E Switch (Private).

or

- 1 x Cisco 2960-S Switch (Management); and
- 1 x Cisco 2960-S Switch (Private).

D.2.2.3.3 Public (Internet) Network

- Address Space: 126.118.70.0/25.

D.2.2.3.4 Management Network

- Address Space: 10.0.0.0/15; and
- DHCP Range: 10.0.5.0-10.1.255.255.

D.2.3 Network Layout

This section summarizes the network configuration for the DAVC Controller, Virtual Host Servers and the Management and Experiment network switches.

D.2.3.1 DAVC Controller (DAVC)

D.2.3.1.1 Public (Internet) Network

Note:

The IP-Address, Subnet mask, Gateway and DNS name servers will most likely be different in your setup than what is indicated below.

- IP Address 126.118.70.18.
- Subnet mask 255.255.255.128.
- Gateway 126.118.70.1.
- DNS name servers 126.118.70.8.

D.2.3.1.2 Management Network

- IP Address 10.0.0.18.
- Subnet mask 255.254.0.0.

D.2.3.2 Virtual Host Servers

D.2.3.2.1 Management Network

- d1: IP Address 10.0.0.101/15, Gateway 10.0.0.18, DNS name servers 10.0.0.18.
- d2: IP Address 10.0.0.102/15, Gateway 10.0.0.18, DNS name servers 10.0.0.18.

D.2.4 Network Switches Configuration

D.2.4.1 Management Network Switch

Be sure to do the following:

- Remove all VLANs except for default; and
- Set ports to access.

D.2.4.2 Private (Experiment) Network Switch

Be sure to do the following:

- Remove all VLANs except for the default; and
- Set ports to trunk (requires incoming and outgoing traffic to be VLAN tagged).

D.2.4.3 Cisco 3750-E Configuration Instructions

Notes:

Connect to the switch using serial console, telnet, HTTP, or SSH to execute the following commands:

```
Switch> enable
Switch# configure   terminal
Switch# vlan 300-999
Switch# end
Switch# show vlan

Switch# configure  terminal
Switch# interface  range  g4/0/1-48
Switch#  switchport  trunk  encapsulation  dot1q
Switch#  switchport mode trunk
Switch# end
Switch# show  interfaces  status
Switch#  write
```

There is a limit to the number of Spanning Tree Protocol instances that can run at once. The VLANs get created, but it seems that only 128 VLANs can be in use with STP running.

D.2.5 DAVC Controller Base Configuration

This section covers the steps needed to configure the DAVC Controller.

D.2.5.1 Install Operating System

D.2.5.1.1 Install Ubuntu 14.04 Server 64-bit

Do the following:

- Select a minimal install;
- Set the hostname to ‘davc’; and
- Configure Automatic Updates.

D.2.5.1.2 Configure User/Group Accounts

Configure User/Group accounts as normal.

D.2.5.2 Install Required Packages

Execute the following:

```
]# aptitude -P install openssh-client openssh-server build-essential ethtool ntp
dstat  sysv-rc-conf  dnsmasq  syslinux  nfs-kernel-server   libdrmaa-dev libapache2-
mod-wsgi python-setuptools xfsprogs python-pip  ubuntu-virt
```

D.2.5.3 Create DAVC Group

Create a ‘davc’ group:

```
]# groupadd davc –g 1001
```

D.2.5.4 Configure DAVC Group Permissions

Give the ‘davc’ group the following permissions in /etc/sudoers:

```
#DAVC v2.0
%davc ALL=NOPASSWD: /opt/davc2.0/davc/scripts/vmscripts/dnsmasq.sh, /bin/chown,
/bin/chmod, /sbin/tune2fs, /sbin/mkfs*, /usr/bin/uuidgen, /usr/bin/qemu-img*
```

D.2.5.5 Create DAVC Directories

Create the following directories:

```
]# mkdir -p /opt/davc2.0
]# mkdir -p /home/PIDs/VHDs
]# chown -R root:davc /home/PIDs
]# chmod 775 -R /home/PIDs
```

D.2.5.6 Install Django 1.7 and Dependencies

```
]# pip install Django==1.7.1
]# pip install django_mathfilters
]# pip install django_bootstrap3==6.2.2
]# pip install djangorestframework
]# pip install django-progressbarupload
]# pip install netaddr
]# pip install ipcalc
]# pip install minixsv
]# apt-get install python-libvirt
```

D.2.5.7 Extract DAVC Package

```
]# tar -xvf davc_<version>.tar.gz --directory /opt/davc2.0
]# chown  -R  root:davc  /opt/davc2.0
]# chmod 775 -R /opt/davc2.0
```

D.2.6 Network Configuration

Note:

Make sure the subnet mask is the same for the management network on the interfaces and DNSMASQ.

D.2.6.1 Configure Public and Management Interfaces

Note:

The IP-Address, Subnet mask, Gateway and DNS name servers should be the same as the configurations in Section D.2.3.

```
]# cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how  to  activate  them. For more information, see interfaces(5).

# auto start
auto lo eth0 eth1

# The loopback network interface
iface lo inet loopback

# Public Network
iface eth0 inet static
address 126.118.70.18
netmask 255.255.255.128
gateway 126.118.70.1
dns-nameservers 126.118.70.8
```

```
# DAVC Management Network
iface eth1 inet static
address 10.0.0.18
netmask 255.254.0.0
```

D.2.7 Name Servers

- 127.0.0.1; and
- 126.118.70.8.

D.2.8 Configure Host File

```
]# cat /etc/hosts
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

## Needed if DNSMASQ goes down

# Management Node
10.0.0.18 davc

# Virtual Host Servers
10.0.0.101 d1
10.0.0.102 d2
```

D.2.8.1 Network Time Protocol (NTP)

Configure NTP as normal.

D.2.8.2 TFTP/PXE (Syslinux)

TFTP/PXE is used to network boot specific Operating System images.

D.2.8.3 Configure Base TFTP/PXE

```
]# mkdir -p /tftpboot/pxelinux.cfg
]# cp  -a  /usr/lib/syslinux/*pxelinux.0  /tftpboot/

]# ll /tftpboot/ total 120
-rw-r--r-- 1 root  root 89501 May 20 2011  gpxelinux.0
-rw-r--r-- 1 root  root 26828 May 20 2011  pxelinux.0
drwxr-xr-x 2 root  root 4096 Mar 6  16:24 pxelinux.cfg
```

Set permissions and ownership:

```
]# chown -R root:davc /tftpboot
]# chmod -R 775 /tftpboot
```

D.2.8.4 Create Localboot PXE Configuration

```
]# cat /tftpboot/pxelinux.cfg/localboot
DEFAULT local
PROMPT 0
TIMEOUT 0
TOTALTIMEOUT 0
ONTIMEOUT local
LABEL local
LOCALBOOT 0
```

D.2.9 Network File System (NFS)

NFS is used for the root file systems (NFSROOT) for the network boot images.

D.2.9.1 Create Directories and Exports

```
]# mkdir /nodelog
]# chown -R root:davc /nodelog
]# chmod -R 775 /nodelog
]# cat /etc/exports
/tftpboot      10.0.0.0/15(ro,no_root_squash,subtree_check)
/home         10.0.0.0/15(rw,no_root_squash,no_subtree_check)
/nodelog       10.0.0.0/15(rw,no_root_squash,subtree_check)
```

D.2.9.2 Increase NFS Processes (Count Depends on Number of Active NFS Clients)

```
]# cat /etc/default/nfs-kernel-server
RPCNFSDCOUNT=32
```

D.2.9.3 Restart NFS Server

```
]# service nfs-kernel-server restart
```

D.2.9.4 Verify NFS

```
]# showmount -e
Export list for davc:
/nodelog  10.0.0.0/15
/home      10.0.0.0/15
/tftpboot 10.0.0.0/15
```

D.2.10 DNSMASQ

DNSMASQ provides DHCP, DNS, and TFTP service for all DAVC Cluster Nodes.

D.2.10.1 Configure DNSMASQ

D.2.10.1.1 Create the DAVC DNSMASQ Directory

```
]# mkdir /etc/dnsmasq.d/davc
]# chmod 775 /etc/dnsmasq.d/davc/
```

D.2.10.1.2 Set the DNSMASQ Conf-dir Variable In /etc/dnsmasq.conf

```
]# grep ^[^#] /etc/dnsmasq.conf
conf-dir=/etc/dnsmasq.d
```

D.2.10.1.3 Update the DNSMASQ base-dnsmasq.conf file

```
]# cat /etc/dnsmasq.d/base-dnsmasq.conf
#  /etc/dnsmasq.d/base-dnsmasq.conf
```

**Set 'conf-dir' to the /etc/dnsmasq.d/davc directory previously created.

```
### Add DAVC Directory
conf-dir=/etc/dnsmasq.d/davc
```

**Set 'interface' to the Management network interface on the DAVC Controller.

```
### General Settings (depends on site)
## listen only on this interface
```

```
interface=eth1
bogus-priv cache-size=5000
log-queries
log-dhcp
```

**Set 'server' to DNS Server on the public network.

```
## Hard Code Upstream DNS Server(s)
no-resolv
server=126.118.70.8
```

**Add the following DHCP options, ensure the 'dhcp-option=option:router' option is set to the DAVC Controller's Management network IP address.

```
### DHCP
dhcp-lease-max=5000
dhcp-option=vendor:PXEClient,1,0.0.0.0
dhcp-option-force=208,f1:00:74:7e
dhcp-option=option:router,10.0.0.18
dhcp-boot=pxelinux.0

## Needed for old gPXE/KVM clients
dhcp-no-override
```

**Set the dhcp range for the dhcp clients on the DAVC Cluster Nodes. Set the nework tag to 'management-net'. The DAVC Cluster Nodes dhcp range should begin at an offset to leave space on the Management network for statically addressed DAVC Virtual Host Servers. Here we assume the DAVC Virtual Host Servers will be given static IP addresses prior to 10.0.0.20 and the DAVC Cluster Nodes will receive DHCP configured IP addresses starting at 10.0.0.20 and up to 10.1.255.254.

```
##DHCP Range
dhcp-range=management-net,10.0.0.20,10.1.255.254,static,255.254.0.0,1h
```

**Enable TFTP.

```
## TFTP
enable-tftp
tftp-root=/tftpboot
tftp-max=1000
```

D.2.11 IPTABLES

Ubuntu uses UFW (Uncomplicated Firewall) to manage IPTABLES.

D.2.11.1 Enable UFW

```
]# cat /etc/ufw/ufw.conf
# Set to yes to start on boot.
# If setting this remotely, be sure to add a rule
# to allow your remote connection before starting ufw.
#  Eg: 'ufw allow 22/tcp'
ENABLED=yes

# Please use the 'ufw' command to set the loglevel.
#  Eg:  'ufw logging medium'.
#  See  'man  ufw'  for  details.
LOGLEVEL=low

]# ufw enable
```

D.2.11.2 Add Rules

> **Note:**
> eth0 is the Public interface and eth1 is the Management interface.

```
]# cat /etc/ufw/after.rules
# Don't delete these required lines, otherwise there will be errors
*filter
:ufw-after-input - [0:0]
:ufw-after-output - [0:0]
:ufw-after-forward - [0:0]

#  End required lines
# don't log noisy services by default
-A ufw-after-input -p udp --dport 137 -j ufw-skip-to-policy-input
-A ufw-after-input -p udp --dport 138 -j ufw-skip-to-policy-input
-A ufw-after-input -p tcp --dport 139 -j ufw-skip-to-policy-input
-A ufw-after-input -p tcp --dport 445 -j ufw-skip-to-policy-input
-A ufw-after-input -p udp --dport 67  -j ufw-skip-to-policy-input
-A ufw-after-input -p udp --dport 68  -j ufw-skip-to-policy-input

# don't log noisy broadcast
-A  ufw-after-input -m addrtype --dst-type BROADCAST -j ufw-skip-to-policy-input

### DAVC ###
# Added for DAVC
-A ufw-after-input -m state --state NEW -p tcp --dport 22 -j ACCEPT
-A ufw-after-input -i eth1 -j ACCEPT

# For routing, uncomment the next two lines

#-A ufw-after-forward -i eth0 -o eth1 -m state --state ESTABLISHED,RELATED -j ACCEPT
#-A ufw-after-forward -i eth1 -o eth0 -j ACCEPT
### DAVC ###

# don't delete the 'COMMIT' line or these rules won't be processed
COMMIT
```

D.2.11.3 Enable NAT Routing (if Desired)

D.2.11.3.1 Enable Kernel Forwarding

```
]# cat /etc/ufw/sysctl.conf
net/ipv4/ip_forward=1
```

D.2.11.3.2 Uncomment ufw-after-forward Lines in /etc/ufw/after.rules

```
# For routing, uncomment the next two lines
-A ufw-after-forward -i eth0 -o eth1 -m state --state ESTABLISHED,RELATED -j ACCEPT
-A ufw-after-forward -i eth1 -o eth0 -j ACCEPT
```

D.2.11.3.3 Add Routing Rules to End of after.rules

```
]# cat /etc/ufw/after.rules
### DAVC ###
# Added for DAVC NAT
*nat
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING -s 10.0.0.0/15 -o eth0 -j MASQUERADE COMMIT
### DAVC ###
```

D.2.11.4 Allow BootPS, DNS, DNSMASQ, DAVC (Port 8001), and Gridengine (Ports 6444 and 6445)

```
]# ufw allow bootps
]# ufw allow 53
]# ufw allow 67
]# ufw allow 8001
]# ufw allow 6445
]# ufw allow 6444
```

D.2.11.5 Restart UFW

```
]# service ufw restart
```

D.2.12 Secure Shell (SSH) Client Configuration

D.2.12.1 Turn Off StrictHostKeyChecking for SSH Client

```
]# cat /etc/ssh/ssh_config
*
*
*
StrictHostKeyChecking  no
*
*
*
```

D.2.12.2 Passwordless SSH for Root

D.2.12.2.1 Generate Key

```
]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): Enter passphrase (empty for
no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa. Your public key has been
saved in /root/.ssh/id_rsa.pub.
The key fingerprint is: 19:22:22:22:22:22:22:22:22:22:22:22:3a:fb:c6:51 root@davc The
key's randomart image is:
<randomart image>
```

D.2.12.2.2 Configure Authorized Key

```
]# cd /root/.ssh
]# cat id_rsa.pub >authorized_keys
```

D.2.12.3 Services

Disable unneeded services. Be sure to keep DNSMASQ, UFW, NFS, and SSHD.

D.2.13 Install Mosquitto MQTT (Message Queuing Telemetry Transport)

Mosquitto is the python implementation of the MQTT publish/subscribe framework.

D.2.13.1 Install Python Dependencies for Mosquitto

```
]# apt-get install python-software-properties python-setuptools
```

D.2.13.2 Add Mosquitto Repository

```
]# apt-add-repository  ppa:mosquitto-dev/mosquitto-ppa
]# apt-get update
```

D.2.13.3 Install Mosquitto

```
]# aptitude install mosquitto
]# pip install paho-mqtt
```

D.2.14 Virtual Host Servers Base Configuration

This section covers the steps needed to configure the DAVC Virtual Host Servers.

D.2.14.1 Install Operating System

D.2.14.1.1 Install Ubuntu 14.04 Server 64-bit

Do the following:

- Select a minimal install;
- Set host name to dn (d1 and d2 in this example); and
- Configure Automatic Updates.

D.2.14.1.2 Configure User/Group Accounts

Configure User/Group accounts as normal.

D.2.14.2 Create DAVC Group

Create a ‘davc’ group.

```
]# groupadd davc -g 1001
```

D.2.14.3 Configure DAVC Group Permissions

Give the davc group the following permissions in /etc/sudoers.

```
#DAVC 2.0
%davc ALL=NOPASSWD: /bin/kill, /bin/chown, /bin/chmod, /usr/bin/virt-install*,
/usr/bin/ovs*,
/usr/bin/virsh*, /sbin/brctl*, /sbin/ifconfig*, /usr/sbin/tunctl*, /sbin/ifconfig*,
/etc/init.d/apache2 restart, /etc/init.d/libvirt-bin*, /usr/sbin/service libvirt-bin*
```

D.2.14.4 Create DAVC Directories

Create the DAVC home directory.

```
]# mkdir -p /opt/davc2.0
```

Create the DAVC image directory – it can be a directory or a mount point.

```
]# mkdir -p /davc/{backups,images,blocks,vnc}
]# mkdir -p /davc/images/repository/VHDs
```

Set the DAVC directory’s owner and permission.

```
]# chown -R root:davc /davc
]# chmod 775 -R /davc
```

D.2.14.5 Extract DAVC Package

```
]# tar -xvf davc_<version>.tar.gz --directory /opt/davc2.0
]# chown  -R  root:davc  /opt/davc2.0
]# chmod 775 -R /opt/davc2.0
```

D.2.14.6 Install Required Packages

Execute the following:

```
]# aptitude -P install openssh-client openssh-server build-essential ubuntu-virt
ethtool ntp dstat sysv-rc-conf nfs-common uml-utilities kvm-ipxe x11-apps vinagre
openvswitch-switch xfsprogs curl python-pip
```

D.2.14.7 Compile Libvirt 0.10.0+ from Source

Install pre-reqs from source:

```
]# apt-get install build-essential libyajl-dev libyajl2 libxml2 libxml2-dev
libdevmapper1.02.1 libdevmapper-dev libnl-3-dev libnl-route-3-dev pkg-config
libgnutls-dev libpciaccess-dev
```

Run the configuration script, compile and install all utilities and libraries using the standard system paths:

```
]# cd /opt/davc2.0/libvirt
]# ./configure --prefix=/usr --localstatedir=/var --sysconfdir=/etc
]# make && make install
```

D.2.14.8 Freeze/Hold Libvirt System Packages

Ensure the libvirt system packages don’t get overwritten in the future by marking them on hold.

```
]# apt-mark hold libvirt-bin
]# apt-mark hold libvirt0
]# apt-mark hold python-libvirt
```

D.2.14.9 Update the System

Note:
You may need to reboot the system after the upgrade has completed.

```
]# aptitude update
]# aptitude -P upgrade
```

D.2.14.10 Configure KVM link

Make sure that the KVM script points to the correct version of qemu.

```
]# cat /usr/bin/kvm
#!/bin/sh exec qemu-system-86_64 -enable-kvm "$@"
```

D.2.15 Network Configuration

D.2.15.1 Configure Host File

Note:
Be sure to remove any 127.0.x.x entry that points to the host name.

```
]# cat /etc/hosts
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

## Needed if DNSMASQ goes down

# Management Node
10.0.0.18 davc

# Virtual Host Servers
10.0.0.101 d1
10.0.0.102 d2
```

D.2.15.2 Host Server Network Interface/Bridge Allocation

Figure D-55 shows an example network interface bridge allocation layout for a host server with 9 physical network interfaces. Each physical interface will be mapped to either a Management or Experiment network bridge. Interfaces mapped to a management bridge are connected to the management switch. Interfaces mapped to an experiment bridge are connected to the experimentation switch.

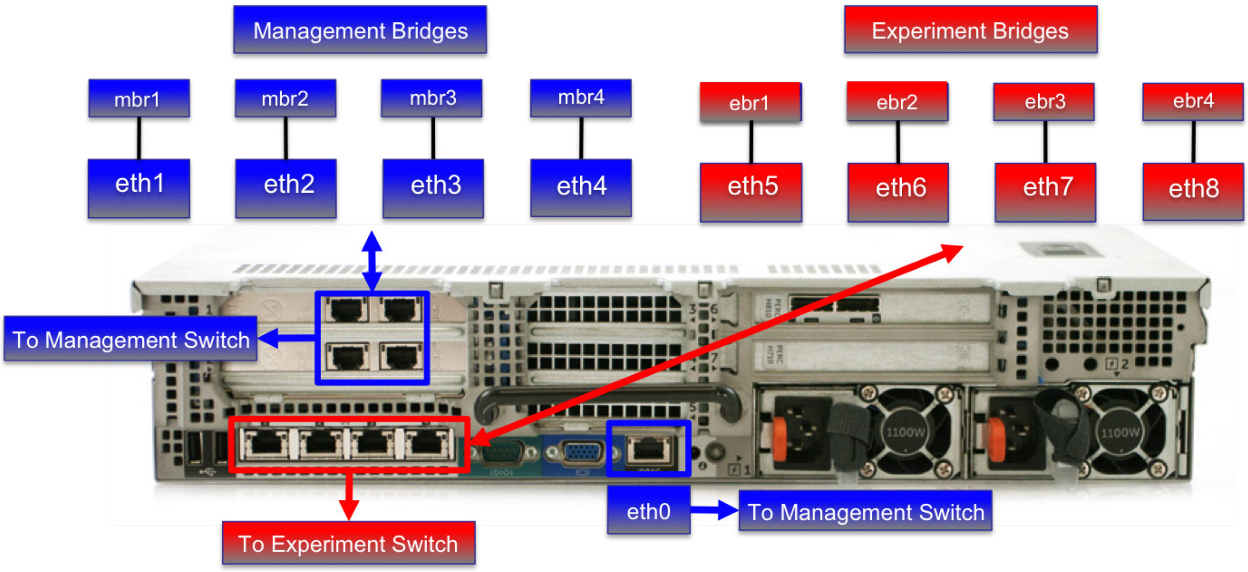

Figure D-55: Host Server Network Interface Bridge Allocation Layout.

D.2.15.2.1 Configure Base Management Interface

```
]# cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# auto start
auto lo eth0

# The loopback network interface
iface lo inet loopback

# DAVC Management Network
iface  eth0   inet  static
```

```
address  10.0.0.101
netmask 255.254.0.0
gateway 10.0.0.18
dns-nameservers 10.0.0.18
```

D.2.15.2.2 Update /opt/davc2.0/server/davc.config

This file sets values for the DAVC Controller server IP (DAVC_SERVER_IP) and port information (DAVC_SERVER_PORT), the names of the Management (MBRIDGE) and Experiment bridges (EBRIDGE) configured for this host, and the VNC proxy server's Public IP Address (DAVC_VNC_PROXY_IP) and Management IP Address (DAVC_VNC_PROXY_CONTROL_IP).

```
DAVC_SERVER_IP=davc
DAVC_SERVER_PORT=8001
DAVC_VNC_PROXY_CONTROL_IP=d1
DAVC_VNC_PROXY_IP=<d1 Public IP Address>
MBRIDGE=mbr0,mbr1,mbr2,mbr3
EBRIDGE=ebr0,ebr1,ebr3,ebr3
```

D.2.15.2.3 Update /opt/davc2.0/server/configure_davc_bridges.sh

Note:
The number of ports for the Management and Private bridges can be different across host servers. Just ensure the appropriate port/bridge mapping is reflected in this script file.

This script configures the bridges for the Management and Experimentation networks. Set MBRIDGE to a comma separated list of the management bridge names on the host server. Set MBRIDGE_PORT to a comma separated list of the corresponding network interfaces associated with each MBRIDGE. Update EBRIDGE and EBRIDGE_PORT in the same manner for the experimentation bridges and network interfaces.

```
MBRIDGE=mbr0,mbr1,mbr2,mbr3
MBRIDGE_PORT=eth1,eth2,eth3,eth4
EBRIDGE=ebr0,ebr1,ebr3,ebr3
EBRIDGE_PORT=eth5,eth6,eth7,eth8
```

If the server does not have a dedicated Management network interface, meaning an interface that is connected to the Management network but not associated with an MBRIDGE, then its Management network IP Address must be associated with one of its MBRIDGEs. If this is the case then the last two lines should read as follows:

```
ifconfig eth0 0.0.0.0
ip addr add 10.0.0.101/255.254.0.0 dev mbr0
```

This IP and interface may be adjusted as appropriate for the DAVC Management network. If the server does have a dedicated control network interface then these lines should be removed.

D.2.15.3 Create Management and Experiment Bridges

Note:
Be sure to verify that a bridge gets assigned to the correct physical port (ethn). For example, if ports eth[1-4] are connected to the Management network, then it does not matter if mbr0 gets assigned to eth4 and mbr2 gets assigned to eth1. For convenience and clarity, we assign them in order.

Run the /opt/davc2.0/server/configure_davc_bridges.sh script. The output of ovs-vsctl show should be similar to below.

```
]# /opt/davc2.0/server/configure_davc_bridges.sh
]# ovs-vsctl show
a9136477-3cdf-49b8-b55d-b3b9eff22e26
Bridge  "mbr0"
   Port "mbr0"
      Interface "mbr0"
      type:  internal
   Port "eth1"
      Interface "eth1"

Bridge "ebr0"
   Port "eth5"
      Interface  "eth5"
   Port "ebr0"
      Interface  "ebr0"
      type: internal

Bridge "ebr3"
   Port "ebr3"
      Interface "ebr3"
      type:  internal
   Port "eth8"
      Interface "eth8"

Bridge  "mbr3"
   Port "eth4"
      Interface  "eth4"
   Port "mbr3"
      Interface "mbr3"
      type: internal

Bridge "ebr1"
   Port "ebr1"
      Interface "ebr1"
      type:  internal
   Port "eth6"
      Interface "eth6"

Bridge  "mbr1"
   Port "mbr1"
      Interface "mbr1"
      type:  internal
   Port "eth2"
      Interface "eth2"

Bridge "ebr2"
   Port "ebr2"
      Interface "ebr2"
      type:  internal
   Port "eth7"
      Interface "eth7"

Bridge  "mbr2"
   Port "mbr2"
      Interface "mbr2"
      type:  internal
   Port "eth3"
      Interface "eth3"

ovs_version: "1.4.6"
```

D.2.16 Disable Multicast Snooping on Private (Experiment) Bridges

D.2.16.1 Create Script to Run at Start Up

```
]# cat /etc/network/if-up.d/disable_multicast-snooping
#!/bin/sh

# Show multicast_snooping setting
for n in 'ls  -d  /sys/class/net/ebr*'
do
```

```
    cat $n/bridge/multicast_snooping
done

# Disable multicast_snooping
for n in 'ls  -d  /sys/class/net/ebr*'
do
    echo 0 >$n/bridge/multicast_snooping
done


# Show multicast_snooping setting
for n in 'ls  -d  /sys/class/net/ebr*'
do
    cat $n/bridge/multicast_snooping
done
```

D.2.16.2 Set Script Permissions

```
]# chmod 755 /etc/network/if-up.d/disable_multicast-snooping
```

D.2.16.3 Restart Networking

```
]# service networking restart
```

D.2.17 SSH Client Configuration

D.2.17.1 Turn Off StrictHostKeyChecking for SSH Client

```
]# cat /etc/ssh/ssh_config
*
*
*
StrictHostKeyChecking  no
*
*
*
```

D.2.17.2 Passwordless SSH for Root

Copy /root/.ssh to /root on Virtual Host Server.

Note:
This step is executed on the DAVC Controller (davc).

```
root@davc:~]# ssh-copy-id d1
```

D.2.18 Mount DAVC Controller NFS

This is for DAVC to be able to forcibly stop/delete jobs, if necessary.

D.2.18.1 Create /Home Entry in /etc/fstab

```
]# cat /etc/fstab
# /etc/fstab: static file system information.
#
# Use blkid to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name
# devices that works even if disks are added and removed. See
# fstab(5).
#
```

```
# <file system> <mount point> <type> <options> <dump> <pass> proc /proc proc
nodev,noexec,nosuid 0 0
# / was on /dev/sda1 during installation
UUID=930ef260-1906-4ff9-a79a-56dd3c0ff395 / ext4 errors=remount-ro 0 1

# swap was on /dev/sda5 during installation
UUID=804584c6-b6b6-4d1c-985a-2efaae90cf1d none swap sw 0 0

davc:/home /home nfs defaults 0 0
```

D.2.18.2 Mount/Home

```
]# mount -a
```

D.2.19 NTP

Configure NTP as normal.

D.2.19.1 Configure Services

Disable unneeded services. Be sure to keep SSHD.

D.2.19.2 Libvirt

D.2.19.2.1 Enable Libvirt Service

```
]# service libvirt-bin start
```

D.2.19.2.2 Disable Libvirt Default Network

```
]# virsh net-destroy default
```

D.2.20 Mosquitto MQTT

Mosquitto is the python implementation of the MQTT publish/subscribe framework.

D.2.20.1 Install Python Dependencies for Mosquitto

```
]# apt-get install python-software-properties python-setuptools
```

D.2.20.2 Add Mosquitto Repository

```
]# apt-add-repository ppa:mosquitto-dev/mosquitto-ppa
]# apt-get update
```

D.2.20.3 Install Mosquitto

```
]# aptitude install mosquitto
]# pip install paho-mqtt
```

D.2.21 DAVC ControllerGrid Engine (ge_master) Configuration

This section covers the steps to configure the Grid Engine Job Scheduler on the DAVC Controller.

D.2.21.1 Install Grid Engine

Install the Grid Engine master and client tools.

```
]# aptitude -P install gridengine-master gridengine-client python-drmaa
```

D.2.21.2 Postfix Configuration

dpkg may ask to configure Postfix. Say no.

```
Postfixno configuration
```

D.2.21.3 Configure Grid Engine Software

D.2.21.3.1 dpkg Portion

dpkg will then ask to configure Grid Engine. Say yes.

```
Configure SGE automatically? Yes cell: default
master hostname: davc

Setting up gridengine-master (6.2u5-3ubuntu1) ...
Initializing cluster  with  the  following  parameters:
=>  SGE_ROOT:  /var/lib/gridengine
=> SGE_CELL: default
=>  Spool  directory:  /var/spool/gridengine/spooldb
=> Initial manager user: sgeadmin
Initializing spool (/var/spool/gridengine/spooldb)
Initializing  global  configuration  based  on  /usr/share/gridengine/default-
configuration
Initializing complexes based on /usr/share/gridengine/centry
Initializing   usersets   based   on   /usr/share/gridengine/usersets Adding user
sgeadmin as a manager
Cluster creation complete
```

D.2.21.3.2 Manual Portion

Set the Grid Engine daemons to start on boot up.

```
]# cat /etc/default/gridengine
# Sun  Grid  Engine  configuration
# Boolean options in this file must be set to yes or no

# Start  the  queue  master  daemon?  (if installed) SGE_START_MASTERD=yes

# Start the execution daemon? (if installed)
SGE_START_EXECD=yes

# SGE_ROOT will default to /var/lib/gridengine SGE_ROOT=/var/lib/gridengine

# SGE_CELL  will default  to  default
SGE_CELL=default
```

Note:

The Ubuntu Grid Engine package places its main configuration files and spooling directories in the following locations:

- /usr/share/gridengine/
- /var/lib/gridengine/

D.2.21.3.3 Export Grid Engine Root

Create file /etc/profile.d/env.sh and add the following line:

```
export SGE_ROOT=/var/lib/gridengine
```

D.2.21.4 Configure Grid Engine Multi-Core Processor Bindings Support

The multi-core processor binding software is located in the DAVC software package tar file in the following directory:

```
/gridengine/sge-hwloc-ssl.tar.gz
```

Unzip the contents of the sge-hwloc-ssl.tar.gz into a temporary folder <tmp>.

```
]# tar -xvf sge-hwloc-ssl.tar.gz
```

Replace the loadcheck file.

```
]# cp <tmp>/utilbin/lx26-amd64/loadcheck /usr/lib/gridengine/
```

D.2.22 Virtual Host Servers Grid Engine (execd) Configuration

This section covers the steps to configure the Grid Engine Job Scheduler on the DAVC Virtual Host Servers.

D.2.22.1 Install Grid Engine

Only install the execd portion of Grid Engine.

```
]# aptitude -P install gridengine-exec
```

D.2.22.2 Postfix Configuration

dpkg may ask to configure Postfix. Say no.

```
Postfixno configuration
```

D.2.22.3 Configure Grid Engine Software

D.2.22.3.1 dpkg Portion

dpkg will then ask to configure Grid Engine. Say yes.

```
Configure SGE automatically? Yes cell: default
master hostname: davc
Setting up gridengine-common (6.2u5-3ubuntu1) ...
Creating config file /etc/default/gridengine with new version
```

D.2.22.3.2 Manual Portion

Set the Grid Engine daemons to start on boot up.

```
]# cat /etc/default/gridengine
# Sun  Grid  Engine  configuration
# Boolean options in this file must be set to yes or no

# Start  the  queue  master  daemon?  (if installed) SGE_START_MASTERD=yes

# Start the execution daemon? (if installed)
SGE_START_EXECD=yes

# SGE_ROOT will default to /var/lib/gridengine SGE_ROOT=/var/lib/gridengine

# SGE_CELL  will default  to  default
SGE_CELL=default
```

Note:

The Ubuntu Grid Engine package places execd files in the following locations:

- /var/spool/gridengine/ (Main Messages)
- /tmp/execd_messages.$PID (Job Error Messages)

D.2.22.4 Configure Grid Engine Multi-Core Processor Bindings Support

The multi-core processor binding software is located in the DAVC software tar file in the following directory: `/opt/davc2.0/gridengine/sge-hwloc-ssl.tar.gz`

Unzip the contents of the sge-hwloc-ssl.tar.gz into a temporary folder <tmp>.

```
]# tar -xvf sge-hwloc-ssl.tar.gz --directory <tmp>
```

Replace the sge_shepherd, sge_execd and loadcheck files.

```
]# service gridengine-exec stop
]# cp <tmp>/bin/lx26-amd64/sge_shepherd /usr/lib/gridengine/
]# cp <tmp>/bin/lx26-amd64/sge_execd /usr/lib/gridengine/
]# cp <tmp>/utilbin/lx26-amd64/loadcheck /usr/lib/gridengine/
```

D.2.23 Install libdrmaa.so.1.0

The library libdrmaa.so.1.0 needs to be installed. Copy the file from the DAVC software installation package to the /usr/lib folder.

```
]# cp /opt/davc2.0/gridengine/libdrmaa.so.1.0 /usr/lib
```

D.2.24 Configure Grid Engine Queue

Note:

Do this on DAVC Controller davc (the Grid Engine Master). All commands can be done as sgeadmin or root.

D.2.24.1 Add Administrative Hosts

Administrative hosts can add, delete, and modify the Grid Engine system.

```
]# qconf -sh davc
]# qconf -ah d1
d1 added to administrative host list

]# qconf -ah d2
d2 added to administrative host list

]# qconf -sh d1
d2 davc
```

D.2.24.2 Add Submit Host

Submission hosts can submit jobs to Grid Engine.

```
]# qconf -as davc
davc added to submit host list
]# qconf -ss davc
```

D.2.24.3 Start Exec Hosts

```
]# ssh d1 -C "service gridengine-exec start"
]# ssh d2 -C "service gridengine-exec start"
```

D.2.24.4 Execution Hosts

Execution hosts run the submitted jobs. They are synonymous with Virtual Host Servers in the DAVC.

D.2.24.4.1 Verify Execution Hosts

Make sure the Virtual Host Servers were added to the execution host list.

```
]# qconf -sel
d1
d2
```

D.2.24.4.2 If Needed, Add Exec Hosts

```
]# qconf -ae d1
]# qconf -ae d2
```

D.2.24.5 Create Queue for All Exec Hosts

Grid Engine execution hosts are grouped within queues. Different queues can be allocated to process different categories of jobs and allow one to perform administrative operations to all of the execution hosts within a queue by referencing the queue name. This step will configure a queue called ‘all.q’ that will contain all execution hosts.

```
]# qconf -aq all.q
root@davc added "all.q" to cluster queue list
```

D.2.24.6 Create Host Group List for all.q

A Host Group List is a way to group all execution hosts. This step configures a host group list with all of the Virtual Host Servers/Execution hosts.

D.2.24.6.1 Show Host Group Lists

```
]# qconf -shgrpl
no host group list defined
```

D.2.24.6.2 Add (Modify) Host Group

To add a new Host Group use the command ’qconf -ahgrp’. To edit an already existing Host Group use the command ‘qconf -mhgrp <host group name>’.

```
]# qconf  -ahgrp
group_name @allhosts
hostlist  d1  d2

root@davc added "@allhosts" to host group list
```

D.2.24.6.3 Show Host Group Lists

```
]# qconf -shgrpl
@allhosts
```

D.2.24.6.4 Show Hosts in Host Group

```
]# qconf  -shgrp
@allhosts group_name
@allhosts hostlist  d1  d2
```

D.2.25 Modify Queue (all.q)

This section covers the steps needed to configure the Grid Engine queue 'all.q'. Note below that entries highlighted in a box are to be updated.

D.2.25.1 Add Host Group to Queue (all.q)

```
]# qconf -mq all.q
qname                     all.q
hostlist                  @allhosts
seq_no                    0
load_thresholds           np_load_avg=1.75
suspend_thresholds        NONE
nsuspend                  1
suspend_interval          00:05:00
priority                  0
min_cpu_interval          00:05:00
processors                UNDEFINED
qtype                     BATCH INTERACTIVE
ckpt_list                 NONE
pe_list                   make
rerun                     FALSE
slots                     8
tmpdir                    /tmp
shell                     /bin/csh
prolog                    NONE
epilog                    NONE
shell_start_mode          posix_compliant
starter_method            NONE
suspend_method            NONE
resume_method             NONE
terminate_method          NONE
notify                    00:00:60
owner_list                NONE
user_lists                NONE
xuser_lists               NONE
subordinate_list          NONE
complex_values            NONE
projects                  NONE
xprojects                 NONE
calendar                  NONE
initial_state             default
s_rt                      INFINITY
h_rt                      INFINITY
s_cpu                     INFINITY
h_cpu                     INFINITY
s_fsize                   INFINITY
h_fsize                   INFINITY
s_data                    INFINITY
h_data                    INFINITY
s_stack                   INFINITY
h_stack                   INFINITY
s_core                    INFINITY
h_core                    INFINITY
s_rss                     INFINITY
h_rss                     INFINITY
s_vmem                    INFINITY
h_vmem                    INFINITY
```

D.2.25.2 Change Load_Thresholds and Slots

Note:
Each Virtual Host Server (execd) may have a different number of slots. The first number listed in the **slots** section is the default value if no entry exists for a specific execd.

```
]# qconf -mq all.q
qname                     all.q
hostlist                  @allhosts
seq_no                    0
load_thresholds           np_load_avg=1
suspend_thresholds        NONE
nsuspend                  1
suspend_interval          00:05:00
priority                  0
min_cpu_interval          00:05:00
processors                UNDEFINED
qtype                     BATCH INTERACTIVE
ckpt_list                 NONE
pe_list                   make
rerun                     FALSE
slots                     8,[d1=16],[d2=24]
tmpdir                    /tmp
shell                     /bin/csh
prolog                    NONE
epilog                    NONE
shell_start_mode          posix_compliant
starter_method            NONE
suspend_method            NONE
resume_method             NONE
terminate_method          NONE
notify                    00:00:60
owner_list                NONE
user_lists                NONE
xuser_lists               NONE
subordinate_list          NONE
complex_values            NONE
projects                  NONE
xprojects                 NONE
calendar                  NONE
initial_state             default
s_rt                      INFINITY
h_rt                      INFINITY
s_cpu                     INFINITY
h_cpu                     INFINITY
s_fsize                   INFINITY
h_fsize                   INFINITY
s_data                    INFINITY
h_data                    INFINITY
s_stack                   INFINITY
h_stack                   INFINITY
s_core                    INFINITY
h_core                    INFINITY
s_rss                     INFINITY
h_rss                     INFINITY
s_vmem                    INFINITY
h_vmem                    INFINITY
```

D.2.25.3 Verify Queue

```
]# qstat -f
queuename     qtype        resv/used/tot.  load_avg    arch         states
-----------------------------------------------------------------------
all.q@d1       BIP             0/0/1           0.01      lx26-amd64
-----------------------------------------------------------------------
all.q@d2       BIP             0/0/1           0.01     lx26-amd64
```

D.2.25.4 Create Queue for KVM Exec Hosts

DAVC maintains queues for each hypervisor within the system. Execution hosts configured to run a particular hypervisor are added to the appropriate queue. This step configures a queue for the KVM hypervisor. In the future, when support is added for other hypervisors, additional queues can be created using these same steps by replacing kvm.q with <hypervisor>.q and @kvmhosts with @<hypervisor>hosts.

```
]# qconf -aq kvm.q
root@davc added "kvm.q" to cluster queue list
```

D.2.26 Create Host Group List for kvm.q

This step configures a host group list with the Virtual Host Servers/Execution hosts configured to run the KVM hypervisor.

D.2.26.1 Show Host Group Lists

```
]# qconf -shgrpl
@allhosts
```

D.2.26.2 Add (Modify) Host Group

To add a new Host Group use the command 'qconf -ahgrp'. To edit an already existing Host Group use the command 'qconf -mhgrp <host group name>'.

```
]# qconf  -ahgrp group_name  @kvmhosts hostlist  d1  d2
root@davc added "@kvmhosts" to host group list
```

D.2.26.3 Show Host Group Lists

```
]# qconf -shgrpl
@allhosts @kvmhosts
```

D.2.26.4 Verify Hosts in Host Group

```
]# qconf -shgrp @kvmhosts group_name
@kvmhosts hostlist  d1  d2
```

D.2.27 Modify Queue (kvm.q)

This section covers the steps needed to configure the Grid Engine queue 'kvm.q'. Note below that entries highlighted in a box are to be updated.

D.2.27.1 Add Host Group to Queue (kvm.q)

```
]# qconf -mq kvm.q
qname                     kvm.q
hostlist          @kvmhosts
seq_no                    0
load_thresholds           np_load_avg=1.75
suspend_thresholds        NONE
nsuspend                  1
suspend_interval          00:05:00
priority                  0
min_cpu_interval          00:05:00
processors                UNDEFINED
qtype                     BATCH INTERACTIVE
ckpt_list                 NONE
pe_list                   make
rerun                     FALSE
```

```
slots                     8
tmpdir                    /tmp
shell                     /bin/csh
prolog                    NONE
epilog                    NONE
shell_start_mode          posix_compliant
starter_method            NONE
suspend_method            NONE
resume_method             NONE
terminate_method          NONE
notify                    00:00:60
owner_list                NONE
user_lists                NONE
xuser_lists               NONE
subordinate_list          NONE
complex_values            NONE
projects                  NONE
xprojects                 NONE
calendar                  NONE
initial_state             default
s_rt                      INFINITY
h_rt                      INFINITY
s_cpu                     INFINITY
h_cpu                     INFINITY
s_fsize                   INFINITY
h_fsize                   INFINITY
s_data                    INFINITY
h_data                    INFINITY
s_stack                   INFINITY
h_stack                   INFINITY
s_core                    INFINITY
h_core                    INFINITY
s_rss                     INFINITY
h_rss                     INFINITY
s_vmem                    INFINITY
h_vmem                    INFINITY
```

D.2.27.2 Change load_thresholds and Slots

Note:

Each Virtual Host Server (execd) may have a different number of slots. The first number listed in the **slots** section is the default value, if no entry exists for a specific execd.

```
]# qconf -mq kvm.q
qname  kvm.q
hostlist                  @kvmhosts
seq_no                    0
load_thresholds           np_load_avg=1
suspend_thresholds        NONE
nsuspend                  1
suspend_interval          00:05:00
priority                  0
min_cpu_interval          00:05:00
processors                UNDEFINED
qtype                     BATCH INTERACTIVE
ckpt_list                 NONE
pe_list                   make
rerun                     FALSE
slots                     8,[d1=16],[d2=24]
tmpdir                    /tmp
shell                     /bin/csh
prolog                    NONE
epilog                    NONE
shell_start_mode          posix_compliant
starter_method            NONE
suspend_method            NONE
resume_method             NONE
terminate_method          NONE
```

```
notify                    00:00:60
owner_list                NONE
user_lists                NONE
xuser_lists               NONE
subordinate_list          NONE
complex_values            NONE
projects                  NONE
xprojects                 NONE
calendar                  NONE
initial_state             default
s_rt                      INFINITY
h_rt                      INFINITY
s_cpu                     INFINITY
h_cpu                     INFINITY
s_fsize                   INFINITY
h_fsize                   INFINITY
s_data                    INFINITY
h_data                    INFINITY
s_stack                   INFINITY
h_stack                   INFINITY
s_core                    INFINITY
h_core                    INFINITY
s_rss                     INFINITY
h_rss                     INFINITY
s_vmem                    INFINITY
h_vmem                    INFINITY
```

D.2.27.3 Verify All Queues

```
]# qstat -f
queuename      qtype          resv/used/tot.  load_avg    arch          states
---------------------------------------------------------------------------------
all.q@d1        BIP              0/0/1            0.01       lx26-amd64
---------------------------------------------------------------------------------
all.q@d2        BIP              0/0/1            0.01       lx26-amd64
---------------------------------------------------------------------------------
kvm.q@d1        BIP              0/0/1            0.01       lx26-amd64
---------------------------------------------------------------------------------
kvm.q@d2        BIP              0/0/1            0.01       lx26-amd64
```

D.2.28 Modify Custom Complex Attributes

D.2.28.1 Create Custom Complex Attributes

Note:

The attribute **mem_free** is built into Grid Engine, while attributes **custom_mem_free, custom_disk_free**, and **custom_vdisk_free** are custom and have to defined manually.

```
]# qconf -mc
#name              shortcut    type    relop  requestable consumable default urgency
#------------------------------------------------------------------------------------
custom_disk_free    c_df       MEMORY    <=        YES          NO          0        0
custom_mem_free     c_mf       MEMORY    <=        YES          YES         0        0
custom_vdisk_free   c_vdf      MEMORY    <=        YES          YES         0        0
mem_free            mf         MEMORY    <=        YES          NO          0        0
```

D.2.28.2 Configure execd for Each Virtual Host Machine

Note:

Set **custom_mem_free** to Virtual Host Server's RAM minus 2 GB (e.g., 32 GB - 2 GB = 30 GB). Set **custom_vdisk_free** to the size of the hard disk minus at least 20 GB. Each execd may have different values.

D.2.28.3 Configure Virtual Host d1

```
]# qconf -me d1
hostname                  d1
```

```
load_scaling              NONE
complex_values            custom_mem_free=30G,custom_vdisk_free=140G
user_lists                NONE
xuser_lists               NONE
projects                  NONE
xprojects                 NONE
usage_scaling             NONE
report_variables          NONE
```

D.2.28.4 Configure Virtual Host d2

```
]# qconf -me d2
hostname                  d2
```

```
load_scaling              NONE
complex_values            custom_mem_free=30G,custom_vdisk_free=140G
user_lists                NONE
xuser_lists               NONE
projects                  NONE
xprojects                 NONE
usage_scaling             NONE
report_variables          NONE
```

D.2.29 Create Custom Load Sensor

Note:

Do this on the Virtual Host Servers. You can create one sensor script and then copy it to the other systems.

D.2.29.1 Create custom_disk_free.sh load sensor

```
]# cat /var/lib/gridengine/bin/custom_disk_free.sh
#!/bin/bash
#
#  custom_disk_free.sh
#
## Partition where VM images reside
PARTITION=/davc/

## Specify where SGE is installed (not needed for Ubuntu package)
#SGE_ROOT=/var/lib/gridengine
#ARCH='$SGE_ROOT/util/arch'
#HOST='$SGE_ROOT/utilbin/$ARCH/gethostname -name'

## For Ubuntu Grid Engine Package, use system hostname HOST='hostname'
end=false
while [ $end = false ]; do
   #      ----------------------------------------
   # wait for an input
   #
   read input result=$?
   if [ $result != 0 ]; then
      end=true
      break
   fi

   if [ "$input" = "quit" ]; then
   end=true break
   fi

   #      ----------------------------------------
```

```
    # send mark for begin of load report echo "begin"

    DFOUTPUT="'df -k $PARTITION| tail -1 | awk 1{ print $4 }1'"
    echo "$HOST:custom_disk_free:${DFOUTPUT}k"

    #      ----------------------------------------
    # send mark for end of load report
    echo "end"
  done
```

D.2.29.2 Set Ownership and Permissions

```
]# chmod 744 /var/lib/gridengine/bin/custom_disk_free.sh
]# chown sgeadmin:sgeadmin /var/lib/gridengine/bin/custom_disk_free.sh
```

D.2.29.3 Copy to All Other Virtual Host Servers

Note:
Make sure that permissions and ownership are correct.

D.2.29.4 Add Custom Load Sensor into Global System

Note:
Do this on the DAVC Controller. Do not change any other line except for the **load_sensor** line.

```
]# qconf -mconf global
load_sensor   /var/lib/gridengine/bin/custom_disk_free.sh
```

D.2.29.5 Verify Custom Load Sensors

```
]# qstat -F custom_mem_free,custom_disk_free,slots,mem_free,custom_vdisk_free
queuename     qtype         resv/used/tot.  load_avg    arch          states
--------------------------------------------------------------------------------
all.q@d1       BIP           0/0/16       0.01          lx26-amd64
hl:mem_free=31.046G
hl:custom_disk_free=154.870G
hc:custom_mem_free=30.000G
hc:custom_vdisk_free=140.000G
qc:slots=16
--------------------------------------------------------------------------------
all.q@d2      BIP            0/0/24       0.01          lx26-amd64
hl:mem_free=31.050G
hl:custom_disk_free=154.805G
hc:custom_mem_free=30.000G
hc:custom_vdisk_free=140.000G
qc:slots=24
```

D.2.30 Test Grid Engine System

D.2.30.1 Create Simple Job Script

```
]$ cat simple.sh
#!/bin/sh
#
# request Bourne shell as shell for job
#$  -S  /bin/sh
#
#$ -N SIMPLE
#
# merge stdout and stderr
#$ -j yes
```

```
# print date and time
date
hostname

# Sleep for 20 seconds
sleep 20

# print date and time again
date
```

D.2.30.2 Submit Simple Job Script

```
]$ qsub simple.sh
Your job 1 ("SIMPLE") has been submitted
```

D.2.30.3 Verify Simple Job Script is Running

```
]$ qstat -f
queuename      qtype          resv/used/tot.  load_avg     arch          states
---------------------------------------------------------------------------------
all.q@d1        BIP              0/0/16            0.03     lx26-amd64
1      0.50000     SIMPLE    root           r            12/04/2012 15:28:01   1
```

D.2.31 DAVC Controller Configuration

This section covers the steps needed to configure the Apache2 Web Server that will host the DAVC web application.

D.2.31.1 Configure DAVC Software on DAVC Controller

D.2.31.1.1 Configure Apache2 Web Server

> **Note:**
>
> This installation assumes the default Apache user 'www-data' will be used. It may be necessary to update the 'www-data' user's password. An alternate Apache user could be used, if so replace the 'www-data' with the appropriate user and substitute the appropriate home directory.
>
> DAVC uses the Django Web Application framework and runs within the Apache2 Web Server.

D.2.31.1.2 Configure Apache2 User

Update the Apache user 'www-data' shell.

```
]# usermod www-data -s /bin/bash
```

Add the Apache user 'www-data' to the 'davc' group.

```
]# usermod -a -G davc www-data
```

Create the Apache user 'www-data' home directory and set its permissions.

```
]# mkdir /var/www
]# chown -R www-data:www-data /var/www
]# chmod 775 /var/www
```

Create .ssh folder and generate ssh keys

```
]# su www-data
]# cd /var/www/
]# mkdir -p .ssh
```

```
]# chmod 700 .ssh/
]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/var/www/.ssh/id_rsa):
Created directory 1/var/www/.ssh1.
Enter passphrase (empty for no passphrase): Enter same passphrase again:
Your identification has been saved in /var/www/.ssh/id_rsa.
Your public key has been saved in /var/www/.ssh/id_rsa.pub.
```

D.2.31.1.3 Install Apache2 Mod WSGI

```
]# apt-get install libapache2-mod-wsgi
]# a2enmod wsgi
```

D.2.31.1.4 Update the Apache2 Configuration

Update the /etc/apache2/httpd.conf file as shown below. Note that (\) indicates the continuation of a line.

```
Alias /static/admin /usr/local/lib/python2.7/dist-
packages/django/contrib/admin/static/admin
Alias /static/js/progress_bar.js  /usr/local/lib/python2.7/dist- \
packages/progressbarupload/static/js/progress_bar.js
Alias /static/jquery.js  /opt/davc2.0/davc/src/davcserver/static/jquery.js
Alias /static/client/client.js
/opt/davc2.0/davc/src/davcserver/static/client/client.js
Alias /static/client/createcluster.js
/opt/davc2.0/davc/src/davcserver/static/client/createcluster.js \
Alias /static/client/clonecluster.js \
/opt/davc2.0/davc/src/davcserver/static/client/clonecluster.js
Alias /static/cluster/details.js
/opt/davc2.0/davc/src/davcserver/static/cluster/details.js
Alias /static/system/system.js
/opt/davc2.0/davc/src/davcserver/static/system/system.js
Alias /static/vhd/vhd.js /opt/davc2.0/davc/src/davcserver/static/vhd/vhd.js
Alias /static/blockdisk/blockdisk.js \
/opt/davc2.0/davc/src/davcserver/static/blockdisk/blockdisk.js
Alias /static/provisioning/rmprovisionclientvhd_v2.py \
/opt/davc2.0/davc/src/davcserver/static/provisioning/rmprovisionclientvhd_v2.p
y


<VirtualHost *:8001>
        ServerAlias davc.com
        WSGIScriptAlias / /opt/davc2.0/davc/src/davc/wsgi.py

        WSGIDaemonProcess   davc.com   python-path=/opt/davc2.0/davc/src/davc
        WSGIProcessGroup davc.com
        WSGIPassAuthorization On

        <Directory /opt/davc2.0/davc/src/davc>
           <Files wsgi.py>
              Order deny,allow
              Allow from all
              Require all granted
           </Files>
        </Directory>

        <Directory /static>
           Order deny,allow
           Allow from all
           Require all granted
        </Directory>

        <Directory /usr/local/lib/python2.7/dist-
   packages/django/contrib/admin/static/admin>
           Order deny,allow
           Allow from all
           Require all granted
        </Directory>
```

```
        <Directory  /opt/davc2.0/davc/src/davcserver/static>
           Order deny,allow
           Allow from all
           Require all granted
        </Directory>

        <Directory  /usr/local/lib/python2.7/dist-
   packages/progressbarupload/static/js>
           <Files progress_bar.js>
              Order  deny,allow
              Allow from all
              Require all granted
           </Files>
        </Directory>
</VirtualHost>
```

Update the /etc/apache2/ports.conf file as shown below:

```
# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf
Listen 80
Listen 8001

<IfModule ssl_module>
   Listen 443
</IfModule>

<IfModule mod_gnutls.c>
   Listen  443
</IfModule>
```

Update the /etc/apache2/apache2.conf file as shown below:

```
# Include all the user configurations:
Include httpd.conf
```

D.2.31.1.5 Update Apache2 envars

Update /etc/apache/envvars with DAVC environment variables.

```
#DAVC 2.0 envs
export DAVC_BACKUP_DIR=/davc/backups
export DAVC_DNSMASQ_DIR=/etc/dnsmasq.d/davc
export DAVC_DNSMASQ_ID=managment-net
export DAVC_HOST_NODE_IMAGES_DIR=/davc/images
export DAVC_HOST_REPOSITORY_DIR=/davc/images/repository/VHDs/
export DAVC_KILL_DIR=/home/PIDS
export DAVC_SCRIPTS_DIR=/opt/davc2.0/davc/scripts/vmscripts
export DAVC_SHARE_DIR=/home/PIDS
export DAVC_TFTP_DIR=/tftpboot
export DAVC_VHD_DIR=/home/PIDS/VHDs/
export DAVC_BLOCK_DISK_DIR=/davc/blocks
export SGE_ROOT=/var/lib/gridengine
export DAVC_NET_CONTROLLER=ovs
export DRMAA_LIBRARY_PATH=/usr/lib/libdrmaa.so.1.0
export PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
```

D.2.31.1.6 Install and Configure MySQL

```
]# apt-get install mysql-server python-dev libmysqlclient-dev python-mysqldb
]# pip install mysql-python
```

D.2.31.1.7 Create DAVC Database

```
]# mysql -u root -p
]# mysql> CREATE DATABASE davc CHARACTER SET utf8;
```

D.2.31.1.8 Create DAVC User

```
]# mysql -u root -p
]#mysql> GRANT ALL PRIVILEGES ON davc.* TO 'davc'@'localhost' IDENTIFIED BY
'<password>' WITH GRANT OPTION;
]#mysql> GRANT ALL PRIVILEGES ON davc.* TO 'davc'@'%' IDENTIFIED BY '<password>' WITH
GRANT OPTION;
```

D.2.31.1.9 Update the Django Database Connection Information

Edit /opt/davc2.0/davc/src/davc/settings.py and update the davc user's password to the password set above.

```
 DATABASES = {
     'default': {
         'ENGINE': 'django.db.backends.mysql',
         'NAME': 'davc',
         'USER': 'davc',
         'PASSWORD': '<password>',
         'HOST': 'localhost',    # Or an IP Address that your DB is hosted on
         'PORT': '3306',
     }
}
```

D.2.31.1.10 Create Database Superuser and Database Tables

Create super user:

```
]# cd /opt/davc2.0/davc/src
]# python manage.py createsuperuser
```

Output:

```
Username (leave blank to use 'root'):
Email address:
Password:
Password  (again):
Superuser created successfully.
```

Create databases.

```
]# cd /opt/davc2.0/davc/src
]# python manage.py migrate
```

Output:

```
Operations to perform:
Synchronize unmigrated  apps:  rest_framework,  bootstrap3
Apply all migrations: admin, contenttypes, davcserver, auth, sessions
Synchronizing apps without migrations:
Creating tables... Installing custom SQL...
Installing indexes...
Running migrations:
Applying  contenttypes.0001_initial...  OK
Applying  auth.0001_initial...  OK
Applying  admin.0001_initial... OK
Applying  davcserver.0001_initial...  OK
Applying  davcserver.0002_auto_20141120_0750...   OK
Applying  davcserver.0003_auto_20141219_0811...   OK
Applying  davcserver.0004_auto_20150508_1359...   OK
Applying  davcserver.0005_blockdisk_quota...   OK
Applying  davcserver.0006_auto_20150611_1835...   OK
Applying  davcserver.0007_auto_20150616_1935...   OK
Applying  davcserver.0008_cluster_istempcreationcluster...    OK
Applying  davcserver.0009_auto_20150625_1405... OK
Applying  davcserver.0010_auto_20150625_1409... OK
Applying  davcserver.0011_auto_20150629_1359... OK
Applying  davcserver.0012_cluster_resourcespulled...    OK
Applying  davcserver.0013_auto_20160125_1700... OK
Applying  davcserver.0014_auto_20160208_1425... OK
Applying  sessions.0001_initial...   OK
```

D.2.31.1.11 Add CRON Job to Refresh DAVC

Update /opt/davc2.0/server/start_davc_server.sh and set the root user's password in the curl command to the super user's password created in the previous section:

```
touch /opt/davc2.0/davc/src/davc/wsgi.py
sleep 5
curl -X POST davc:8001/davc/api/server/start/ -u root:<password> -d '{}' -H \
"Content-Type:application/json"
```

Update /etc/crontab to include the following rule:

```
#refresh davc service
00 22 * * * root /opt/davc2.0/server/start_davc_server.sh
```

D.2.31.1.12 Update ProgressbarUpload Python Library to Account for Different JSON Libraries

Perform the copy commands shown below. This will update the ProgressbarUpload python library to use conditional python import statements.

```
]# cp /opt/davc2.0/progressbarupload/views.py /usr/local/lib/python2.7/dist-\
packages/ progressbarupload/.
]# cp /opt/davc2.0/progressbarupload/uploadhandler.py \ /usr/local/lib/python2.7/dist-
packages/ progressbarupload/.
```

D.2.32 Configure DAVC Software On Host Servers

D.2.32.1 Configure Apache2 User

Set the Apache2 user 'www-data' password.

```
]# passwd www-data
```

Update the Apache2 user 'www-data' shell.

```
]# usermod www-data -s /bin/bash
```

Add the Apache 2 user 'www-data' to the 'davc' group and 'libvirtd' group.

```
]# usermod -a -G davc www-data
]# usermod -a -G libvirtd www-data
```

Create the Apache2 user 'www-data' home directory and set its permissions.

```
]# mkdir /var/www
]# chown -R www-data:www-data /var/www
]# chmod 775 /var/www
```

Create .ssh folder.

```
]# su www-data
]# cd /var/www/
]# mkdir -p .ssh
]# chmod 700 .ssh/
```

> **Note:**
> Perform this next step on the DAVC Controller davc as the 'www-data' user.

```
]# cd /var/www/.ssh
]# ssh-copy-id  www-data@<HOSTSERVER>
```

D.2.32.2 Configure Virtual Hard Drive Service (VHD) On Host Servers

The VHD Service is an automated process that copies uploaded VHDs from the DAVC Controller to a local repository on the DAVC Host Server.

D.2.32.2.1 Add the VHD Service to rc.local So it Will Be Started When The Host Servers Boot

The VHD Service Script is located at /opt/davc2.0/hosts/launch_vhdsyncer.sh. The VHD Service script takes the hostname of the DAVC Controller as its only parameter.

Update /etc/rc.local as below:

```
]# cat /etc/rc.local
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By  default  this  script  does  nothing.
/opt/davc2.0/server/config_davc_bridges.sh
/opt/davc2.0/hosts/launch_vhdsyncer.sh davc &
exit   0
```

D.2.32.2.2 Start the VHD Service

```
]# /opt/davc2.0/hosts/launch_vhdsyncer.sh davc > /dev/null &
```

D.2.33 Create A DAVC System Configuration

A DAVC System Configuration defines system-wide settings and constraints for DAVC Clusters and Cluster Nodes.

D.2.33.1 Access DAVC Web Application Login Page

At this point DAVC should be installed and accessible via a web browser at the following URLs (replace <DAVC CONTROLLER MANAGEMENT IP ADDRESS> or <DAVC CONTROLLER PUBLIC IP ADDRESS> with the correct DAVC Controller IP address):

http://<DAVC CONTROLLER MANAGEMENT IP ADDRESS>:8001/davc or

http://<DAVC CONTROLLER PUBLIC IP ADDRESS>:8001/davc

Log in using the form in the upper right hand part of the DAVC Home Page (Figure D-56). Use the superuser username and password created in Section D.2.31.1.10.

D.2.33.2 Create A New System Configuration

After logging in, Click the ’Create System Config’ button in the DAVC System Administration Page (Figure D-57) to access the System Configuration Form.

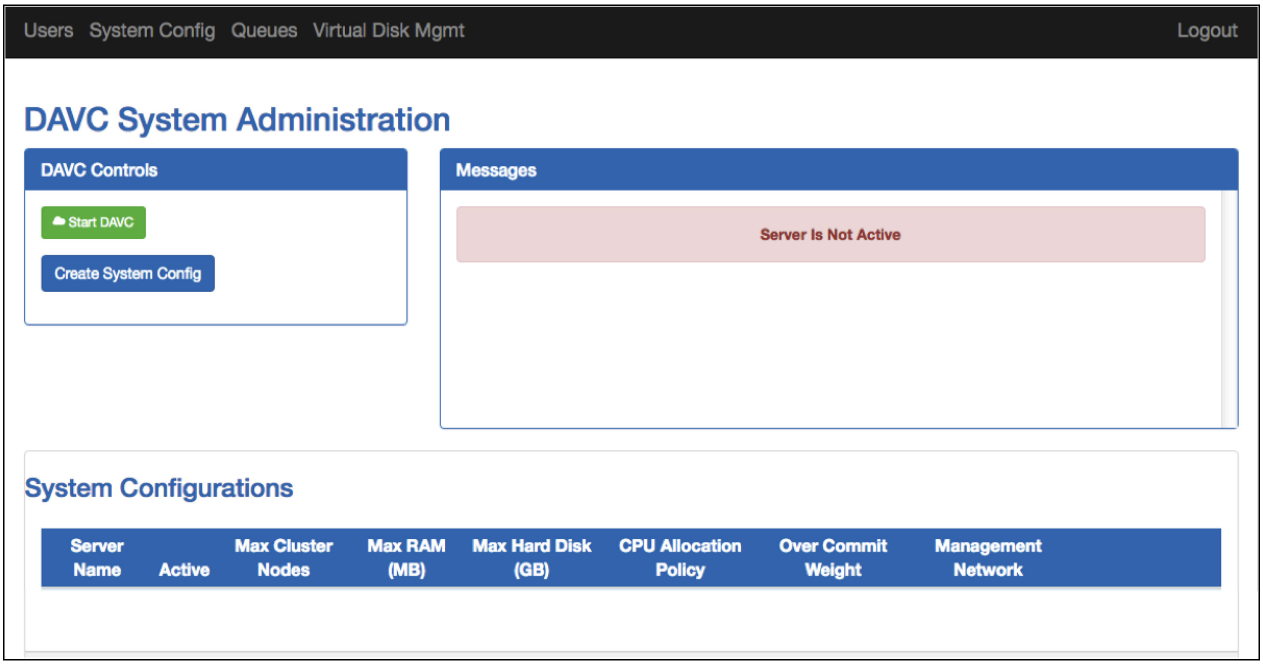

Figure D-56: DAVC Home Page and Login.

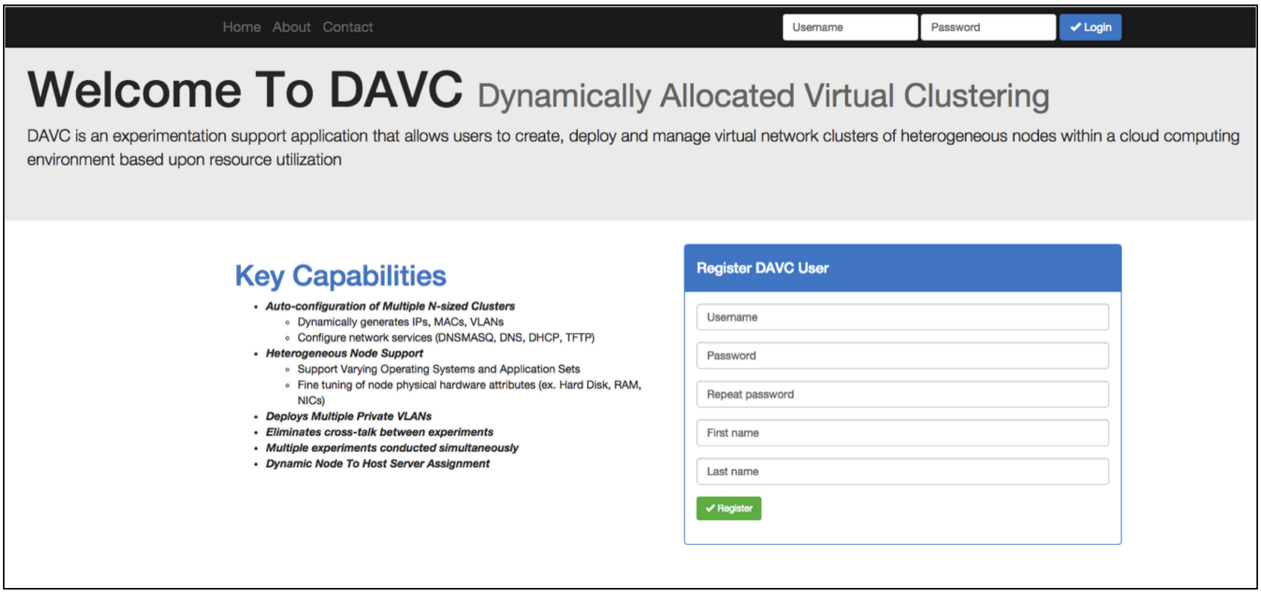

Figure D-57: DAVC System Administration Page.

Edit the System Configuration form (Figure D-58) and fill in the appropriate values for the following:

- DAVC Server Management Network Hostname – This is the DAVC Controller's Management Network hostname.
- Max Node RAM in MB – This is the maximum amount of RAM a Cluster Node can have in MB.
- Max Nodes Per Cluster – This the maximum number of Nodes that can be included in a single Cluster.

- Max Node Hard Drive Size in GB – This is the maximum non-persistent hard disk space a single Cluster Node can be allocated in GB.
- VLAN Pool Range – This defines the range of VLAN IDs the system will use when creating Cluster networks. Each Experiment network added to a cluster will be allocated a VLAN from this pool.
- Node to CPU CORE Allocation Policy – This determines how the system will allocate CPU CORES to Cluster Nodes. The options include:
  - Share CPU CORES – Cluster Nodes share all the CPU CORES on the DAVC Host Server where they are allocated.
  - Do Not Share CPU CORES – All Cluster Nodes will be allocated to dedicated CPU CORES as defined by that Cluster's specific configuration.
  - 1 Node To 1 CPU CORE – All Cluster Nodes will be allocated to a single dedicated CPU CORE ignoring the Cluster's specific configuration.
- Over Commit CPU CORES – This Policy Is Not Implemented.
- Over Commit Weighting – This Value Is Not Used.
- Management Network In CIDR (Classless Inter-Domain Routing) Format – This is the Management Network in CIDR Format.

Click the 'Create Config' button when complete.

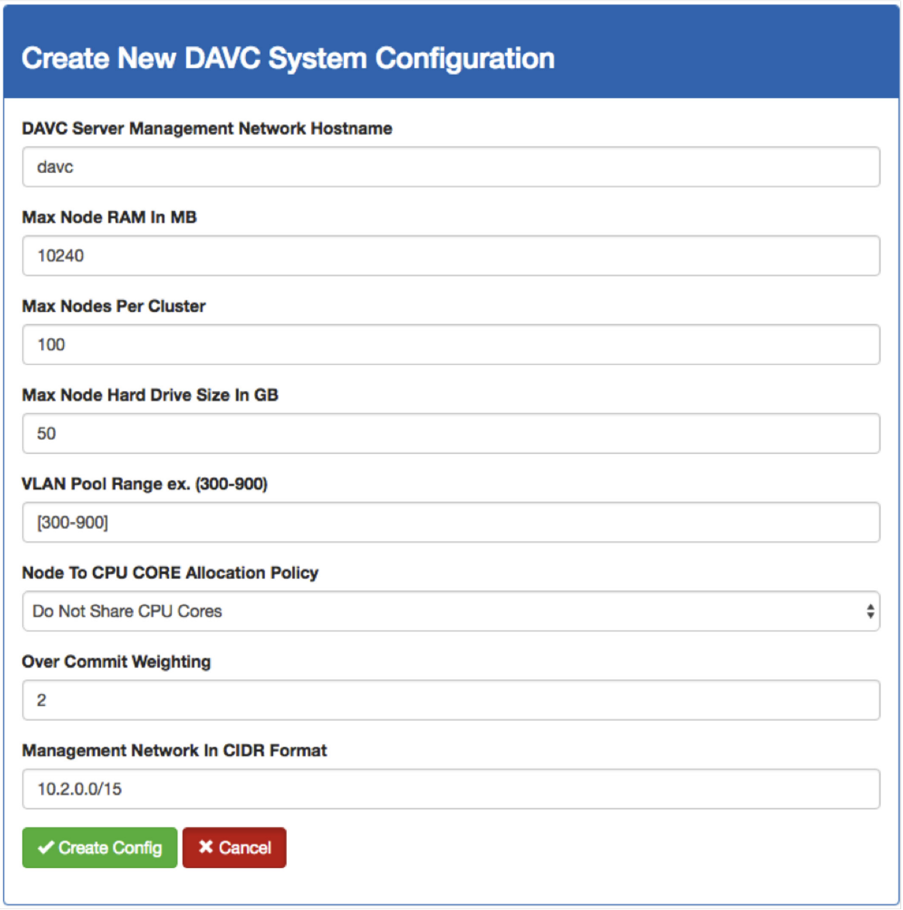

Figure D-58: System Configuration Form.

D.2.33.3 Active the System Configuration

The new System Configuration should now be listed in the DAVC System Administration System Configuration list. Administrator's Console. Click 'Activate' in the 'Config Options' dropdown menu (Figure D-59).

> **Note:**
> The DAVC System Configuration activation process may take a few minutes to complete depending upon the size of the Management network defined in the configuration.

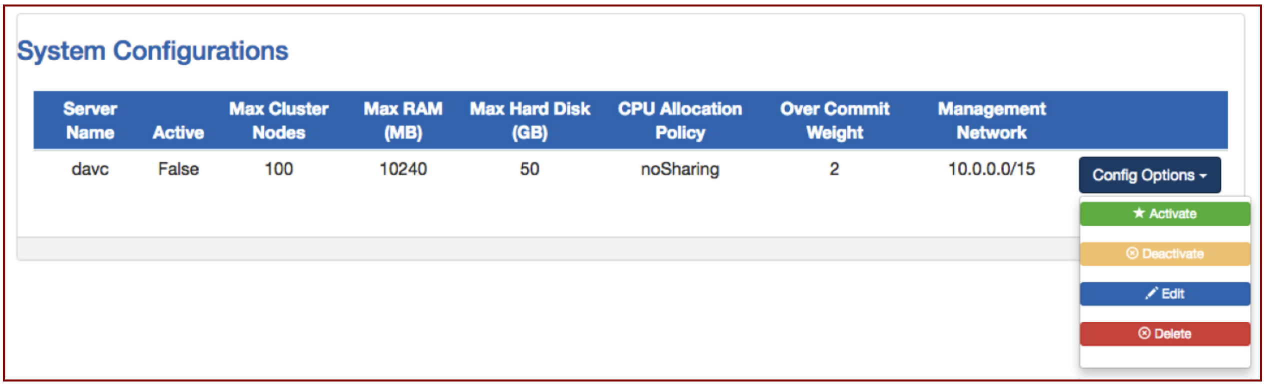

Figure D-59: Activate System Configuration.

D.2.34 Deploy VHD Template Images for DAVC

D.2.34.1 Registering A Virtual Hard Drive In DAVC

DAVC can register/install Virtual Hard Drives (VHDs) that contain preinstalled operating systems. The VHDs can then be used as templates to create cluster nodes.

D.2.34.2 Update the Node Provisioning Startup Client Script to Include Correct Controller Hostname

The DAVC node provisioning startup client script auto configures the VHD on boot-up. The script assumes that the hostname for the controller is 'davc'. This is used as a fallback in the event that the controller hostname cannot be determined automatically. If the hostname for the controller has been changed from the default, the provisioning startup script should be updated accordingly.

```
]# cat /opt/davc2.0/davc/scripts/provisioning/provision_startup.sh

#get the DAVC server address
DAVCSERVER='grep dhcp-server-identifier /var/lib/dhcp/dhclient.eth0.leases |\
uniq | cut echo  $DAVCSERVER
if [ "$DAVCSERVER" == "" ];
then
   echo  "Couldn't Get DAVC Server Address From DHCP Records."  >>  \
   /opt/getProvisioning.log
   echo "Using Default DAVC Server Address: davc" >> \
   /opt/getProvisioning.log
   DAVCSERVER="davc"
fi
```

Modify line 13 of this script (DAVCSERVER="davc") with the correct hostname for the DAVC Controller.

D.2.34.3 Installing the DAVC Node Provisioning Client Startup Script in a Virtual Hard Drive Images

A Virtual Hard Drive Image must be preinstalled with the DAVC Node Provisioning Client Startup Script in order to be compatible with DAVC. This client runs after the virtual machine has booted and performs configurations of NFS, VLANs, NICs and several services including SSH and DNS. The DAVC Node Provisioning Startup Client is located in the following directory along with a wrapper start script:

```
/opt/davc2.0/davc/scripts/provisioning/rmprovisionclientvhd_v2.py
/opt/davc2.0/davc/scripts/provisioning/provision_startup.sh
```

D.2.34.4 Configure Node Provisioning Client Startup Script to Run at Boot

Copy the client and the startup script to Virtual Hard Drive's /opt directory and add an entry to /etc/rc.local so the startup script will be launched when the virtual machine boots up.

```
]# cat /etc/rc.local
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.
/opt/provision_startup.sh
exit 0
```

D.2.34.5 Configure Virtual Hard Drives for Hotplug Support

Hotplug support is required so DAVC Block Disks can be attached and detached to and from the running Virtual Machine without rebooting.

```
]# echo 'acpiphp' >> /etc/modules
]# echo 'pci_hotplug' >> /etc/modules
```

D.2.34.6 Configure Network Interfaces on Virtual Hard Drives

The DAVC Node Provisioning Client Startup Script expects only the interface 'lo' and 'eth0' to be active and configured for DHCP on boot up for Virtual Hard Drives. This can be achieved by editing the network interfaces configuration file (Debian-based) as below:

```
]# cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and  how  to  activate  them.  For more information,  see  interfaces(5).

# The loopback network
interface auto lo
iface lo inet loopback

#  The  primary  network
interface auto  eth0
iface eth0 inet dhcp
Also ensure the persistent network labelling rules file is empty so that interfaces
provisioned by DAVC will be labelled starting with eth0:
]# cat /etc/udev/rules.d/70-persistent-net.rules
# This file was automatically generated by the /lib/udev/write_net_rules
# program, run by the persistent-net-generator.rules rules file.
#
# You can modify it, as long as you keep each rule on a single
# line, and change only the value of the NAME= key.
```

D.2.34.7 Clear Hostname File on Virtual Hard Drive

DAVC provides each virtual machine node with its hostname, so ensure the hostname file is also empty:

```
]# cat /etc/hostname
```

D.2.34.8 Clear DHCP Leases on Virtual Hard Drive

The DAVC server provides DHCP services to each node's control interface (eth0), so ensure any existing DHCP leases are removed:

```
]# rm /var/lib/dhcp/dhclient.eth0*
```

D.3 DYNAMICALLY ALLOCATED VIRTUAL CLUSTERING MANAGEMENT SYSTEM USER'S GUIDE

The Dynamically Allocated Virtual Clustering Management System (DAVC) is an experimentation infrastructure that provides the means to dynamically create, deploy, and manage virtual clusters of heterogeneous nodes within a cloud-computing environment. The system allows researchers to create virtual clusters of nodes that can be used for experimentation, software development, and integration with existing hardware and software. This report provides usage instructions for the DAVC version 2.0 web application.

This report is separated into the following sections, which detail, via examples and step-by-step instructions, actions the user will perform when using DAVC version 2.0:

1) Accessing and logging into DAVC;
2) DAVC cluster configuration;
3) DAVC cluster instantiation;
4) DAVC cluster and node details;
5) DAVC virtual hard disk management;
6) DAVC block disk/persistent storage management; and
7) Creating a new virtual hard disk from a cluster node.

Each section contains slides from a PowerPoint presentation on using DAVC version 2.0. The slides are presented without change from the original version or additional comment.

D.3.1 Accessing and Logging into DAVC

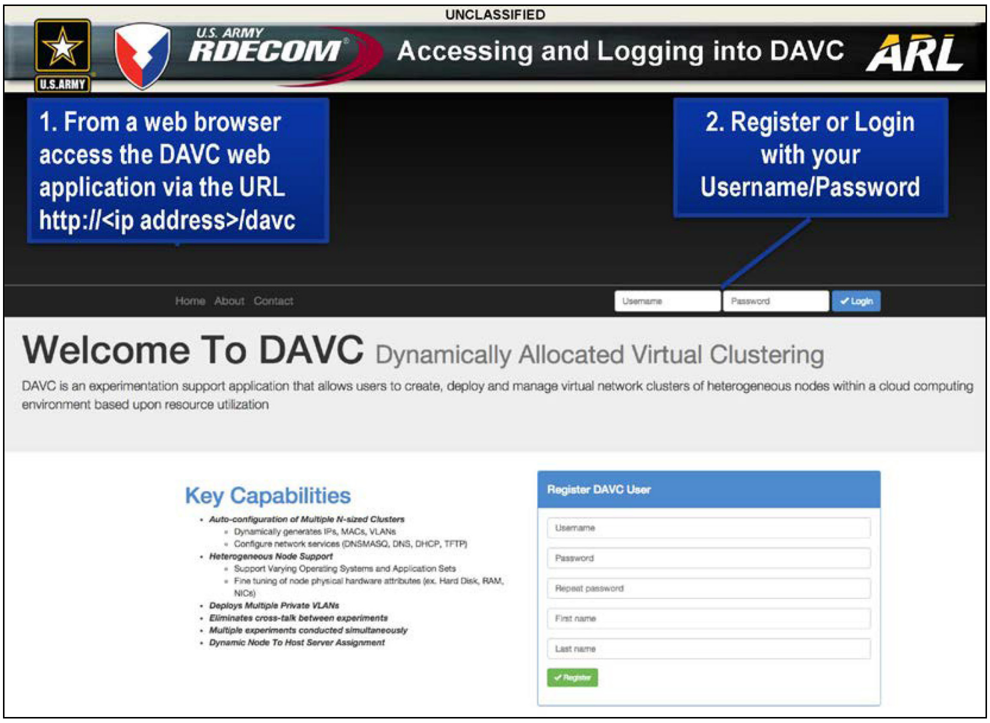

Figure D-60: Accessing and Logging into DAVC.

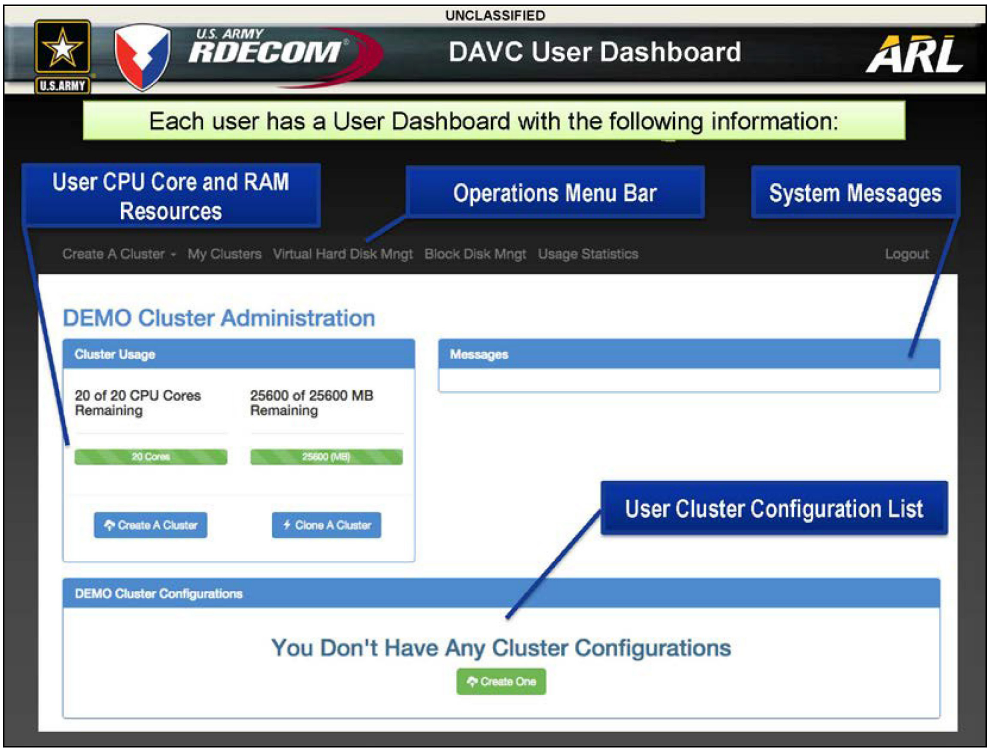

Figure D-61: DAVC User Dashboard.

D.3.2 DAVC Cluster Configuration

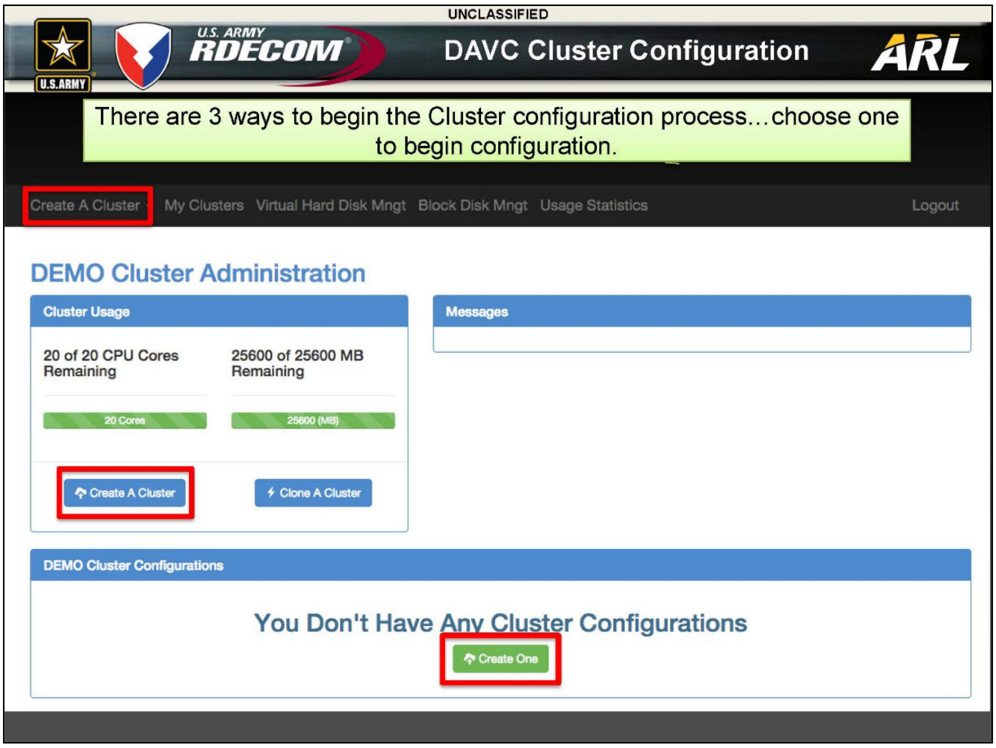

Figure D-62: DAVC Cluster Configuration.

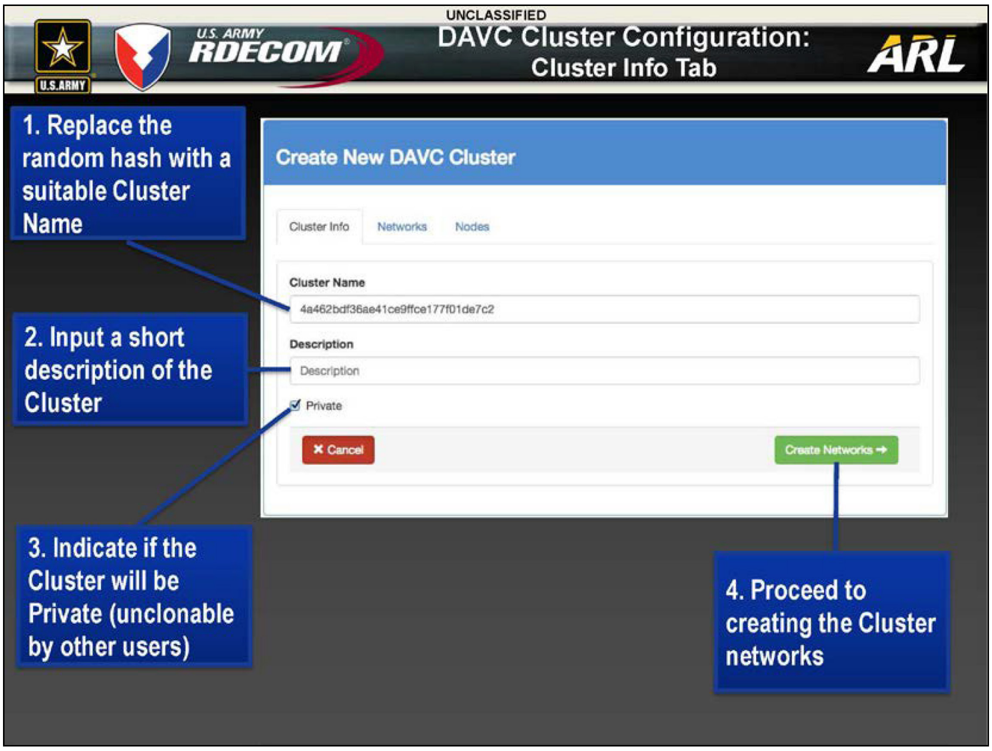

Figure D-63: DAVC Cluster Configuration: Cluster Info Tab.

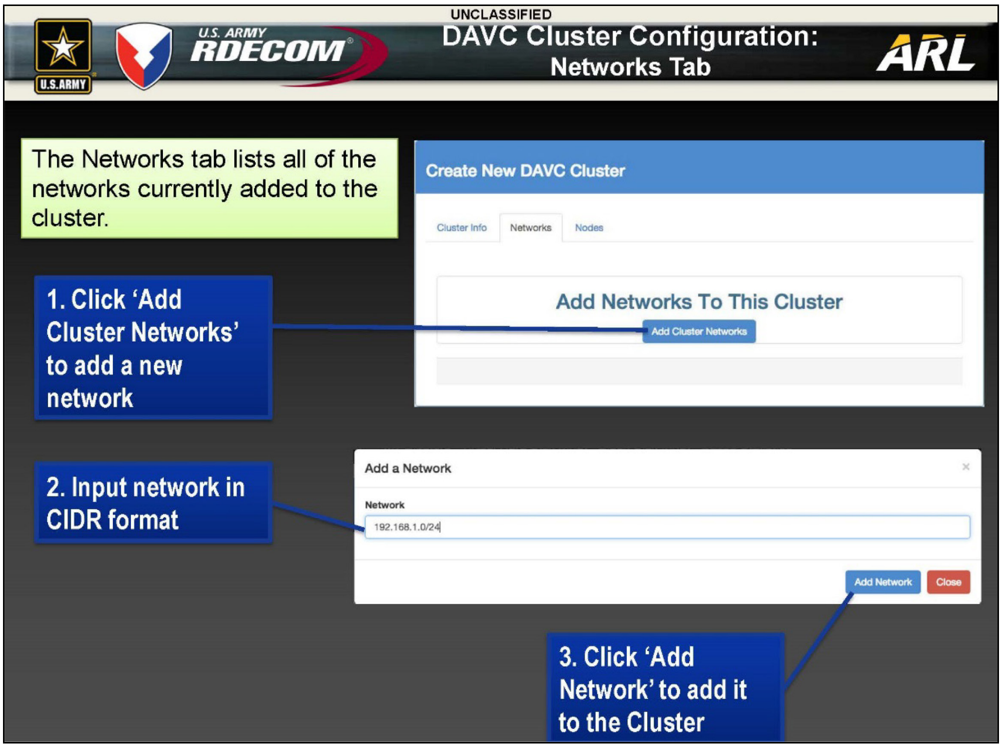

Figure D-64: DAVC Cluster Configuration: Networks Tab.

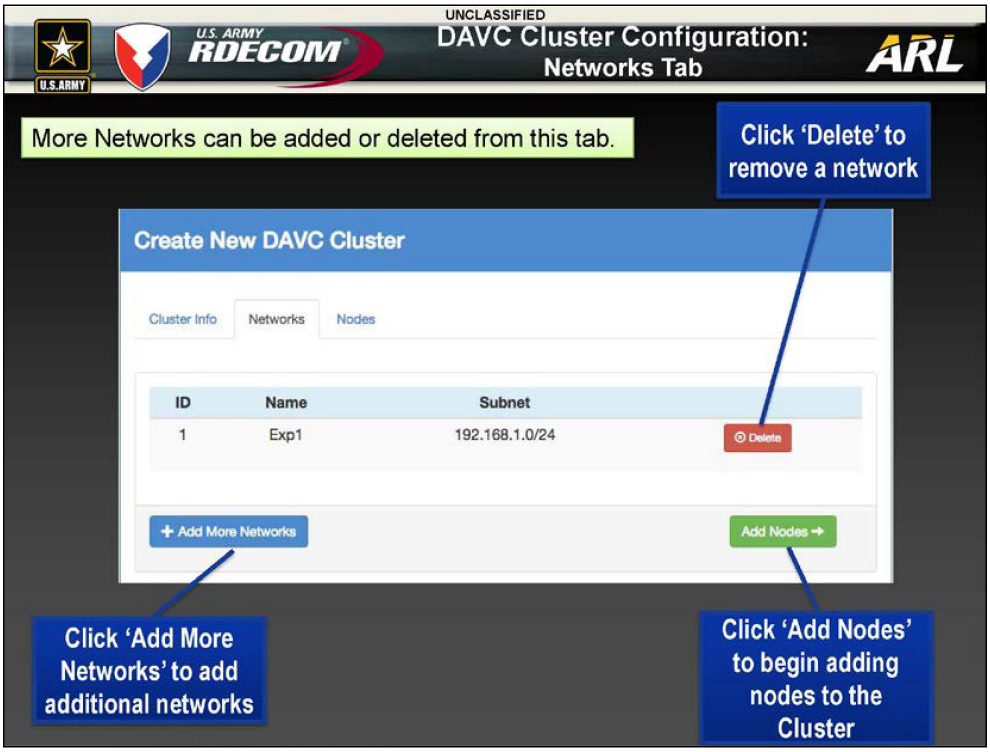

Figure D-65: DAVC Cluster Configuration: Networks Tab.

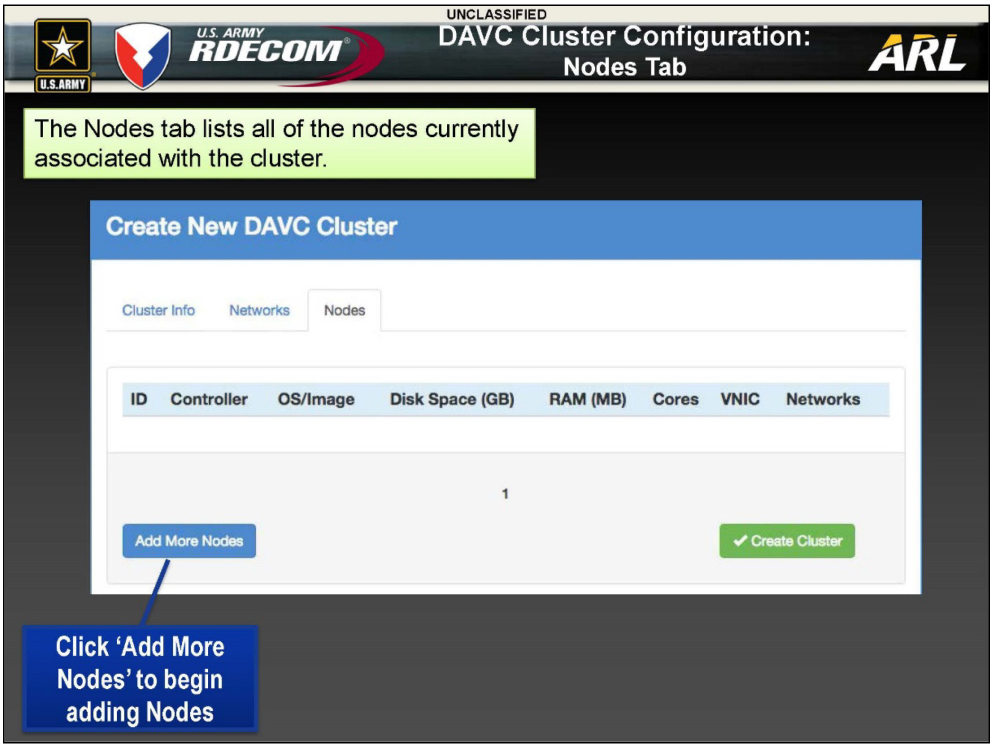

Figure D-66: DAVC Cluster Configuration: Nodes Tab.

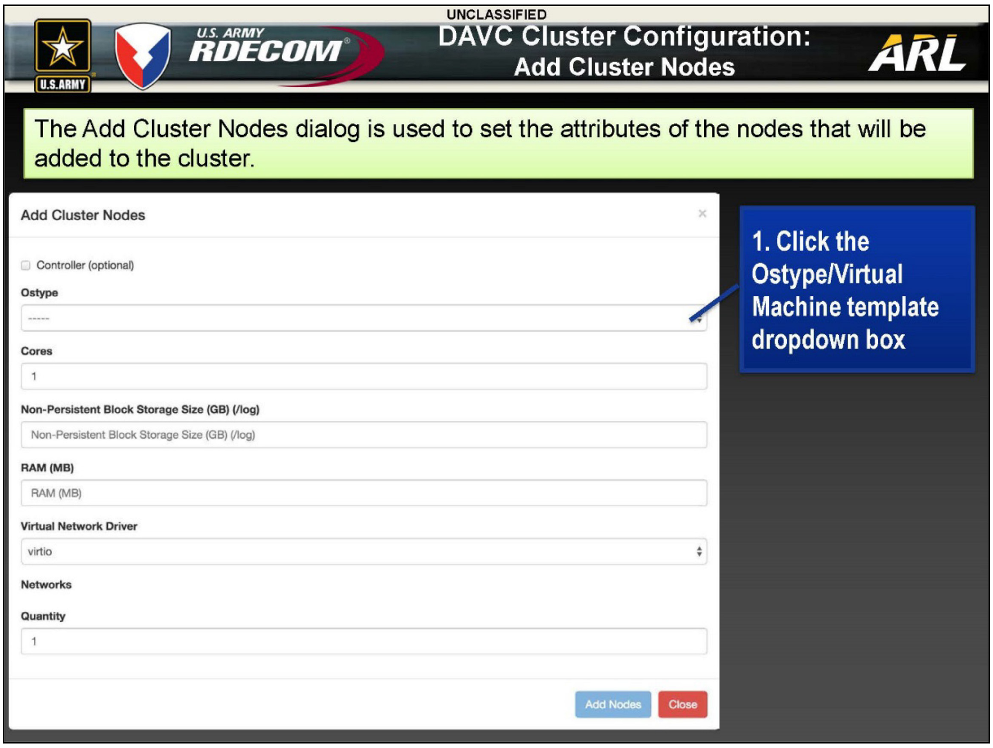

Figure D-67: DAVC Cluster Configuration: Add Cluster Nodes.

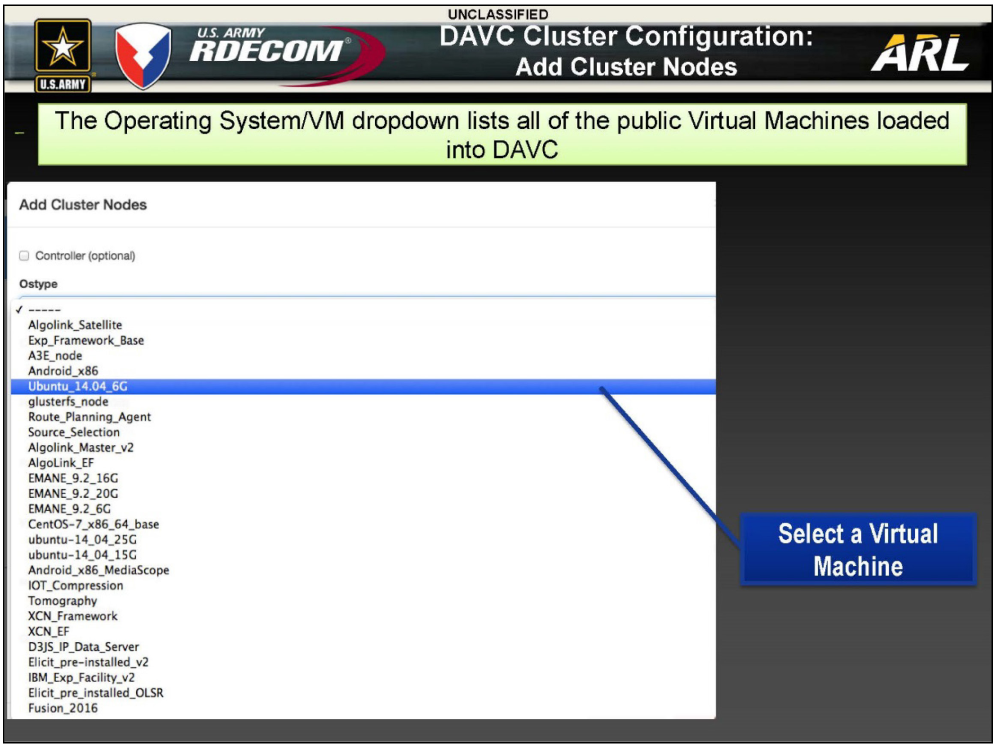

Figure D-68: DAVC Cluster Configuration: Add Cluster Nodes.

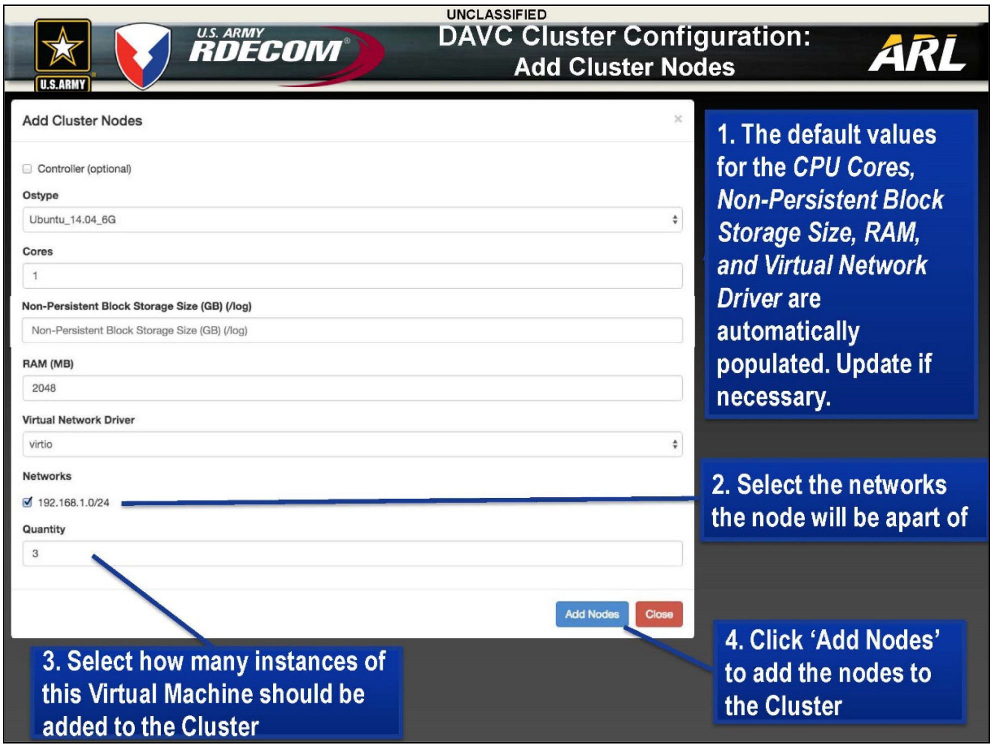

Figure D-69: DAVC Cluster Configuration: Add Cluster Nodes.

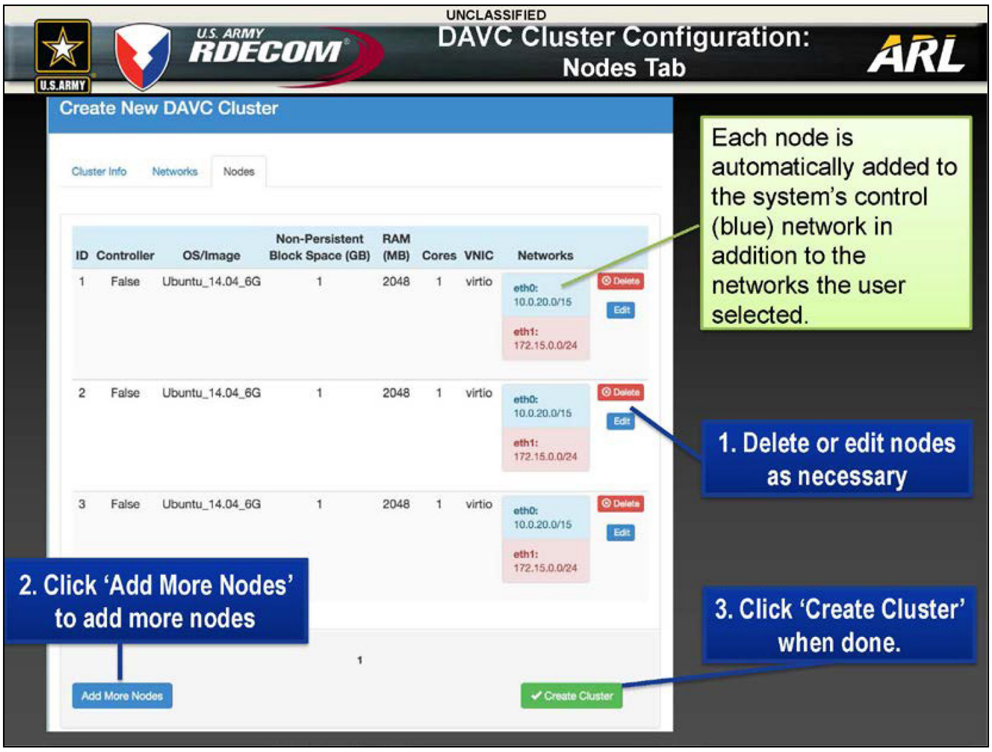

Figure D-70: DAVC Cluster Configuration: Nodes Tab.

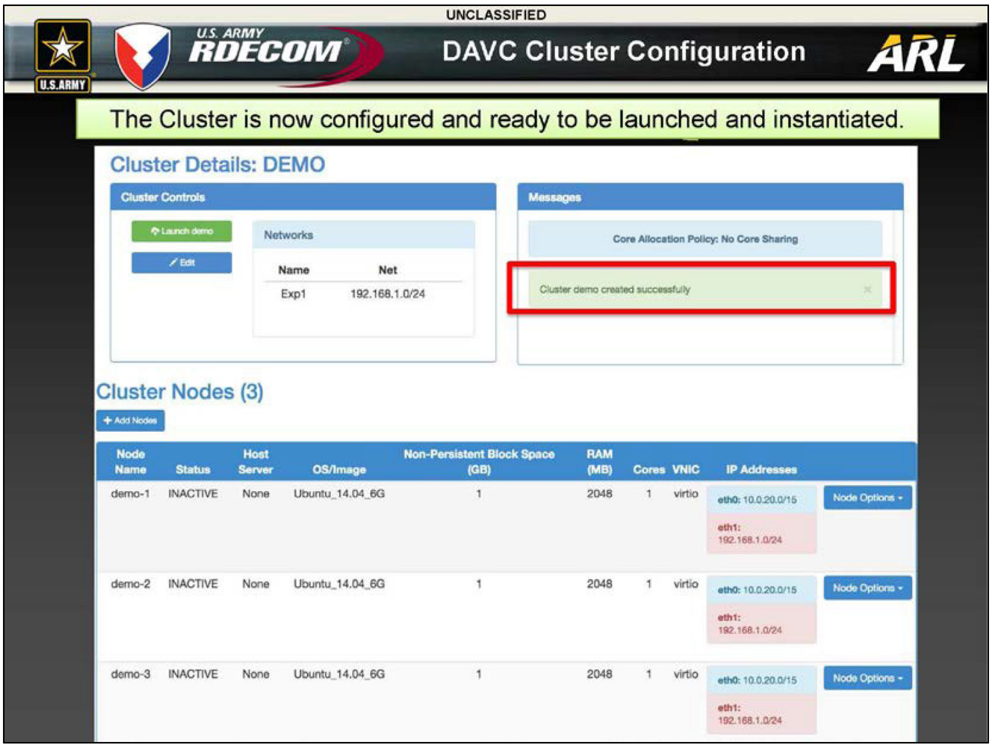

Figure D-71: DAVC Cluster Configuration.

D.3.3 DAVC CLUSTER Instantiation

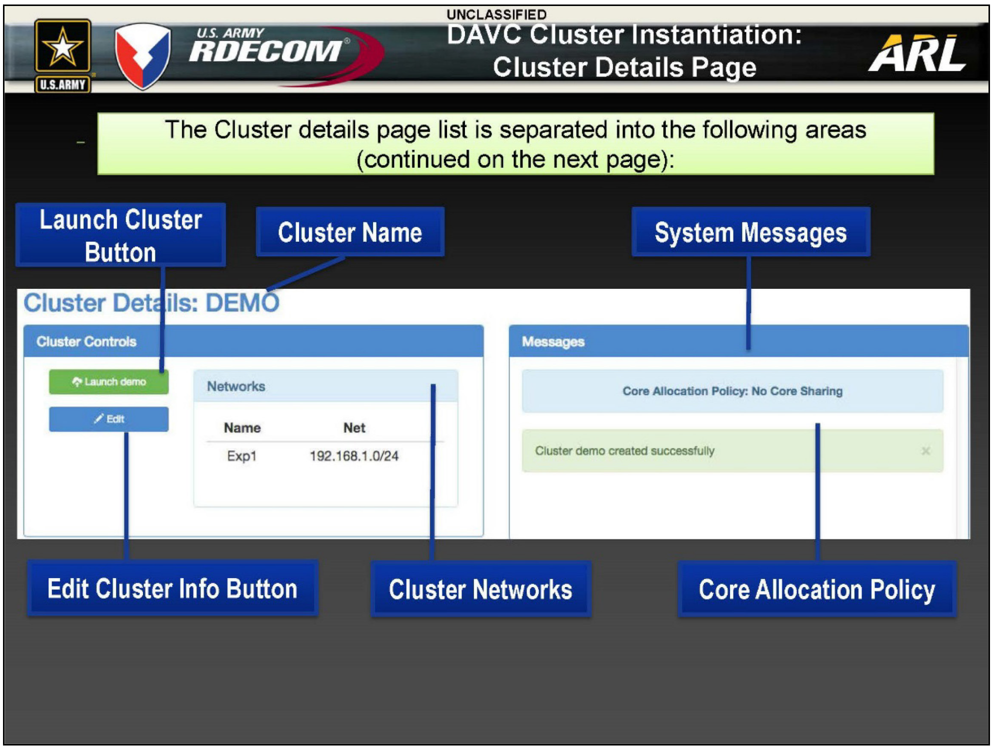

Figure D-72: DAVC Cluster Instantiation: Cluster Details Page.

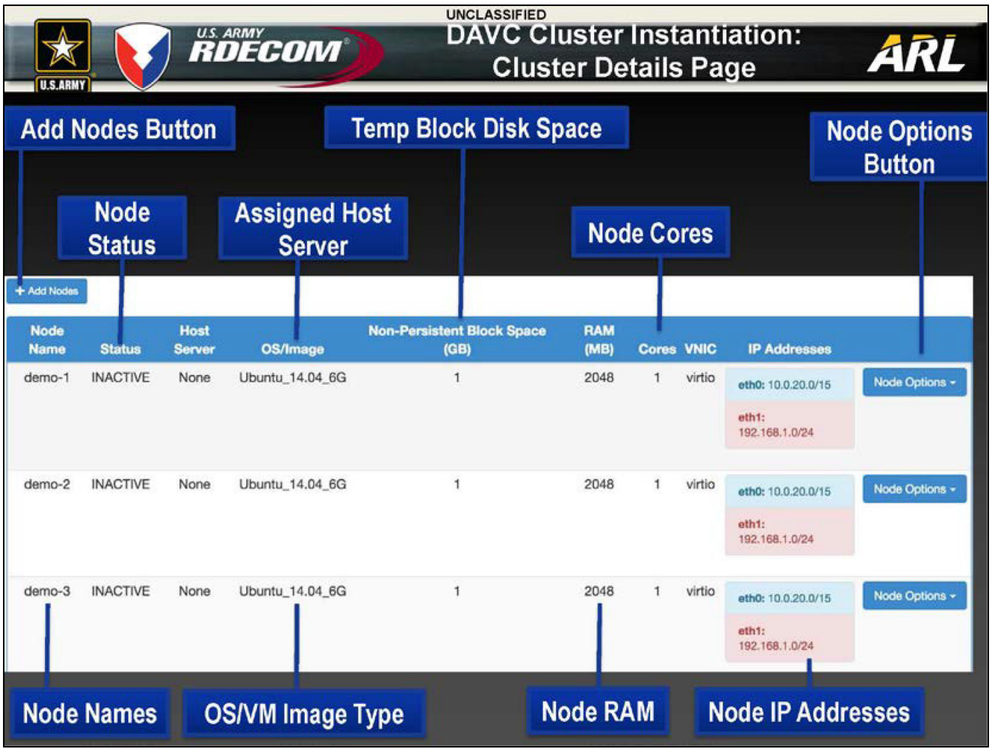

Figure D-73: DAVC Cluster Instantiation: Cluster Details Page.

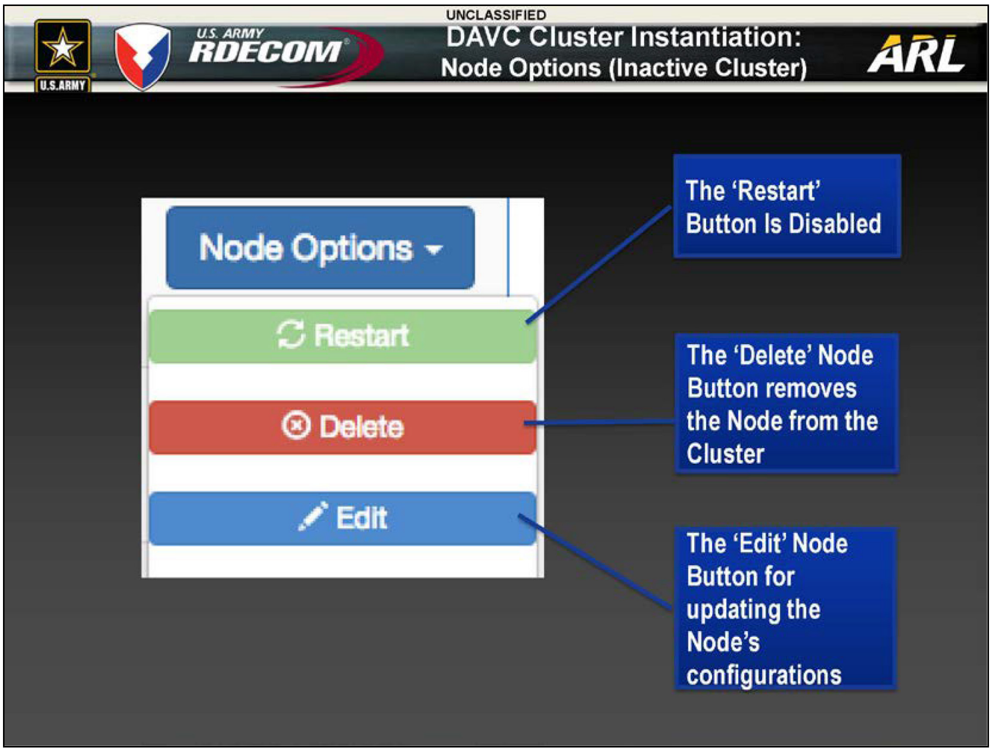

Figure D-74: DAVC Cluster Instantiation: Node Options (Inactive Cluster).

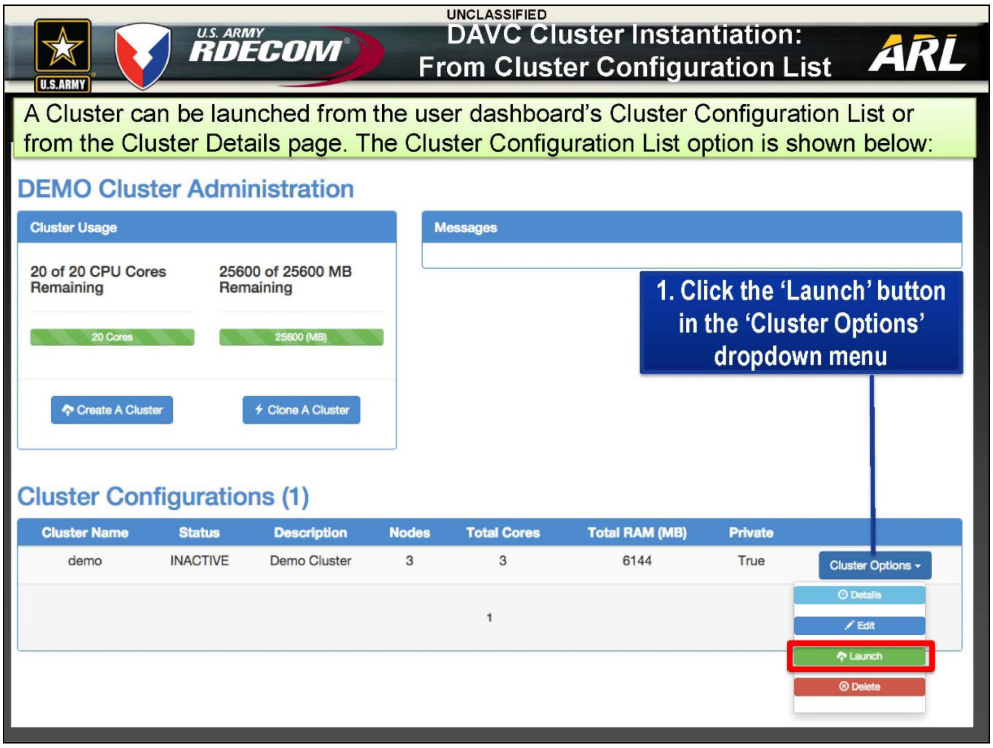

Figure D-75: DAVC Cluster Instantiation: From Cluster Configuration List.

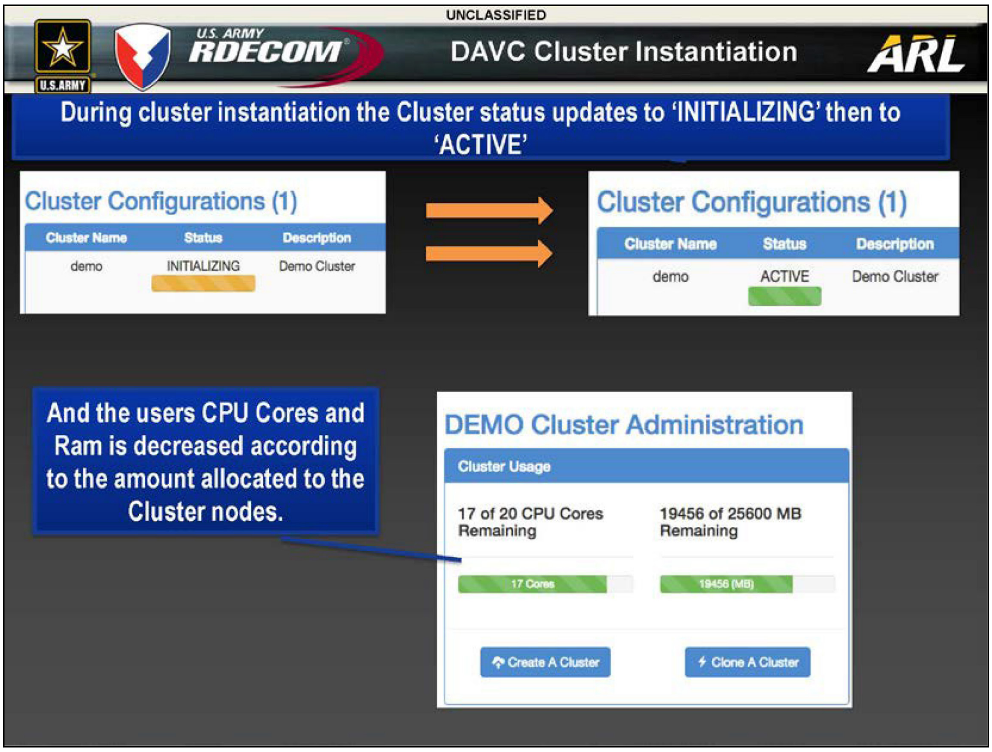

Figure D-76: DAVC Cluster Instantiation.

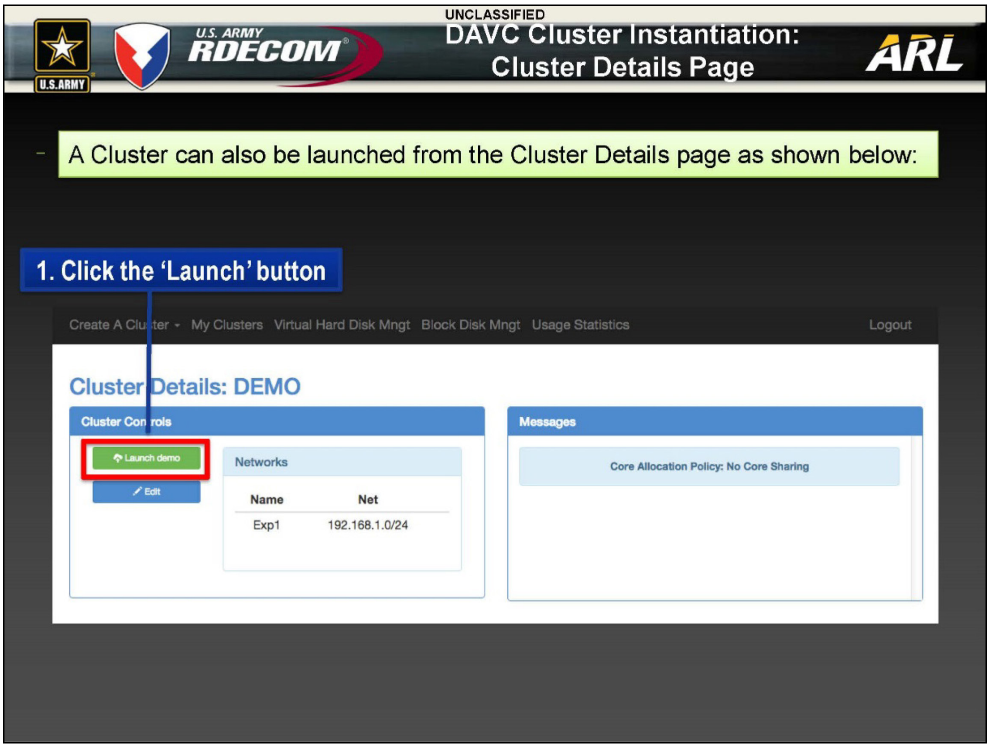

Figure D-77: DAVC Cluster Instantiation: Cluster Details Page.

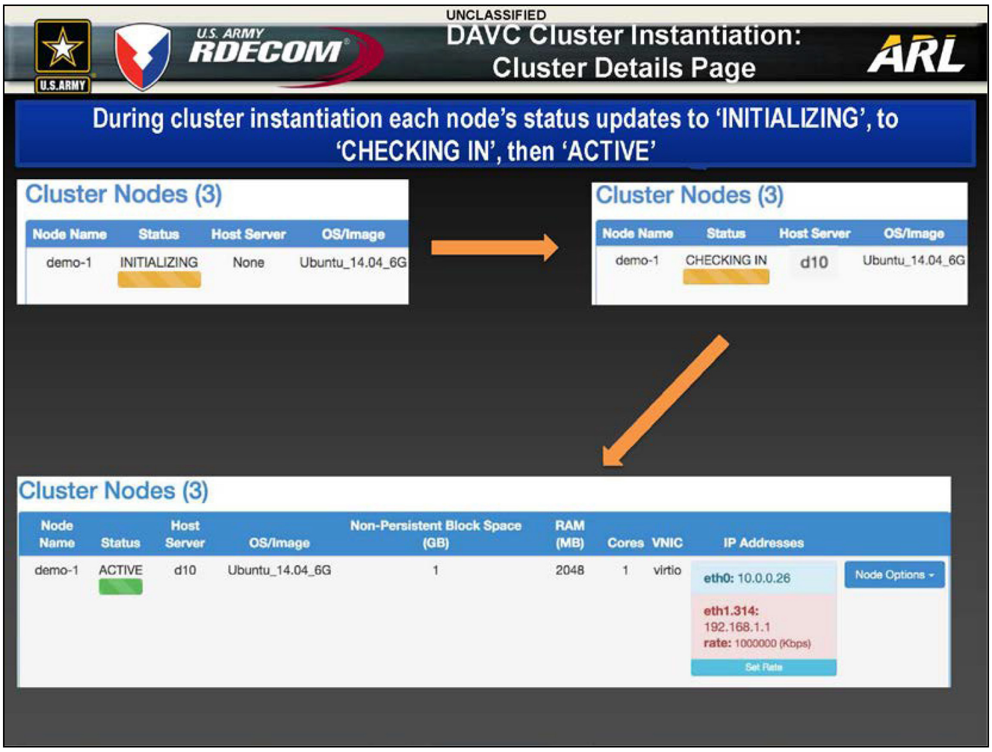

Figure D-78: DAVC Cluster Instantiation: Cluster Details Page.

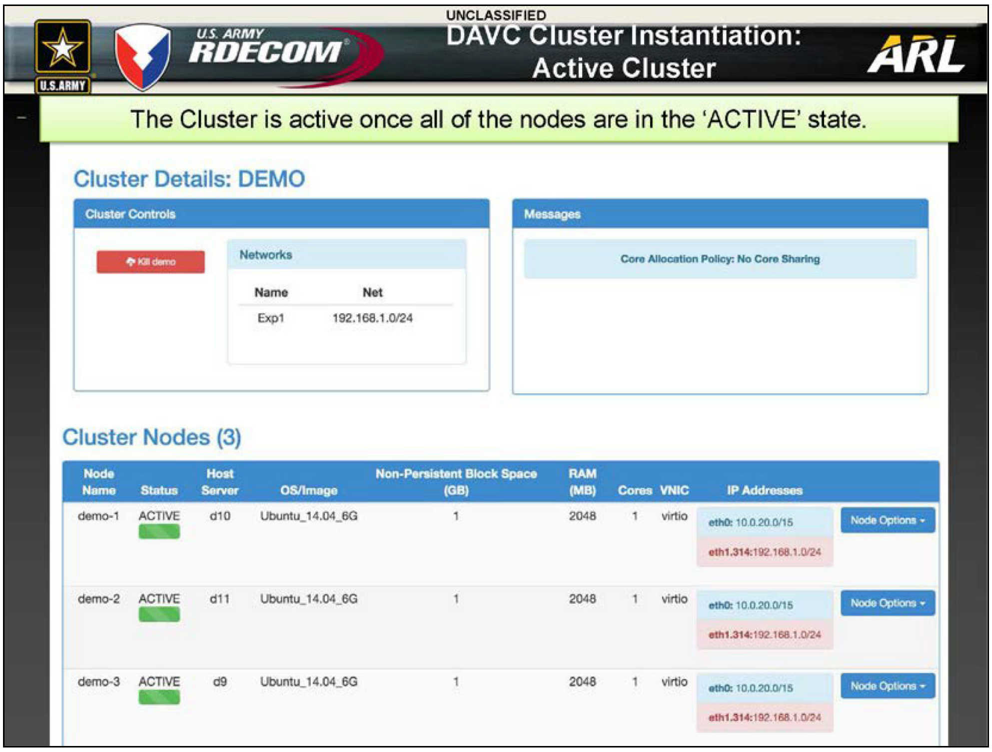

Figure D-79: DAVC Cluster Instantiation: Active Cluster.

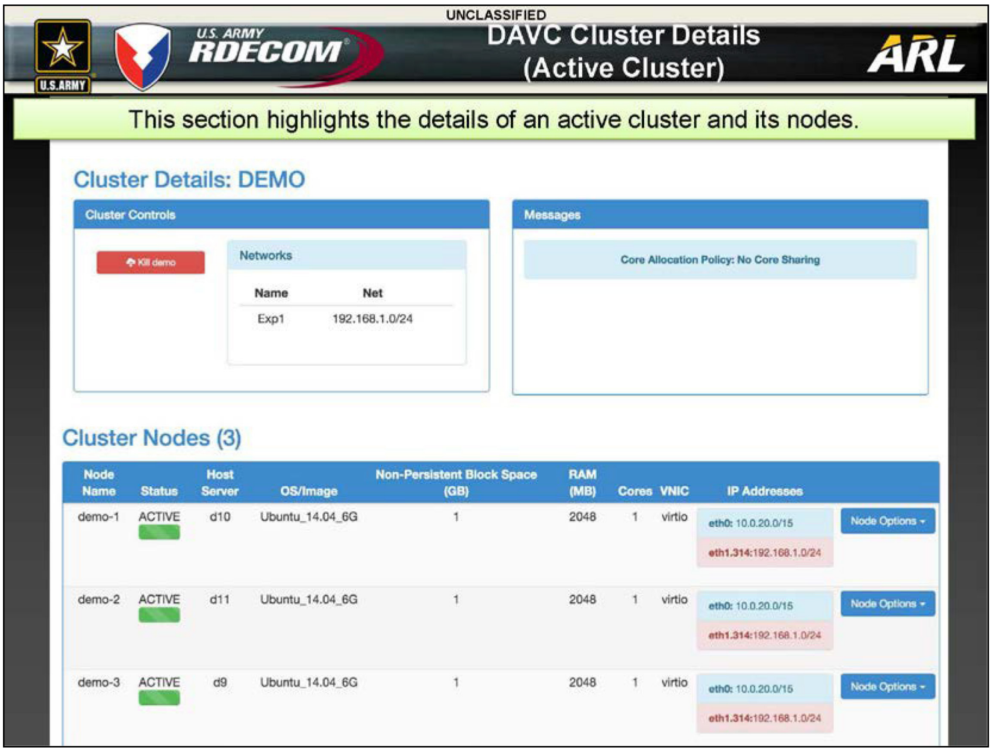

Figure D-80: DAVC Cluster Details: (Active Cluster).

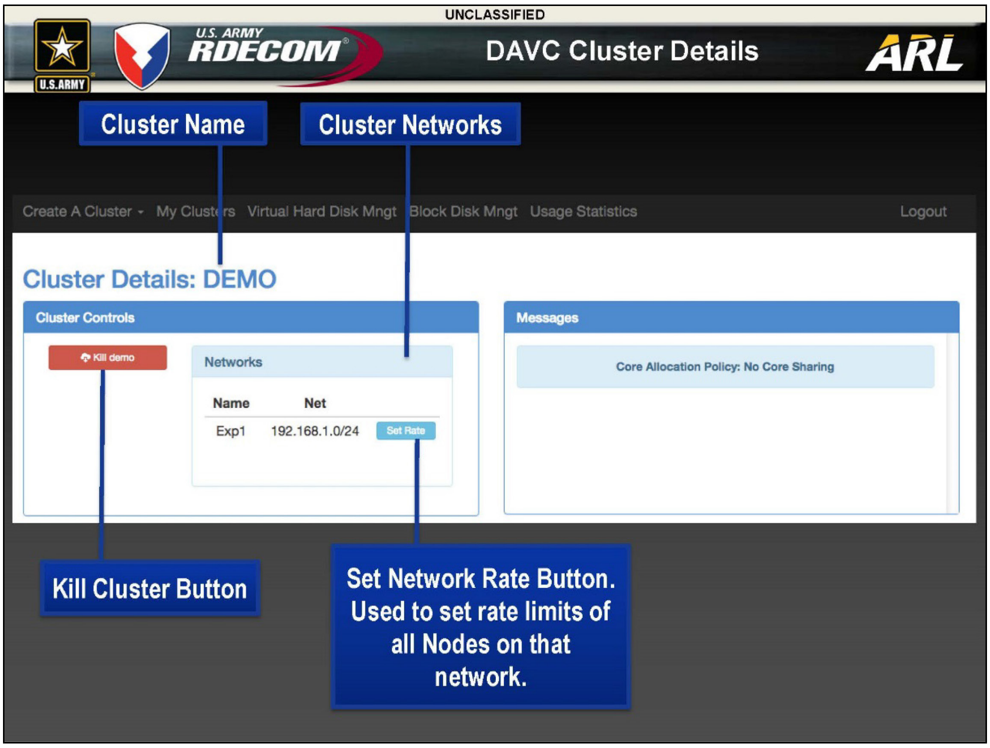

Figure D-81: DAVC Cluster Details.

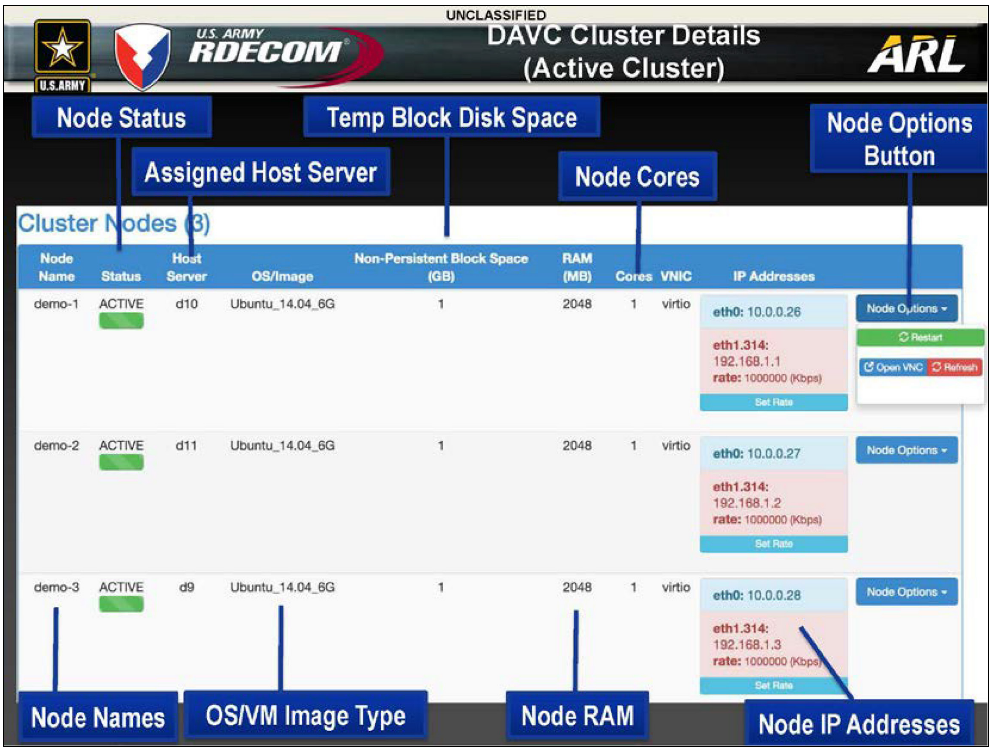

Figure D-82: DAVC Cluster Details: (Active Cluster).

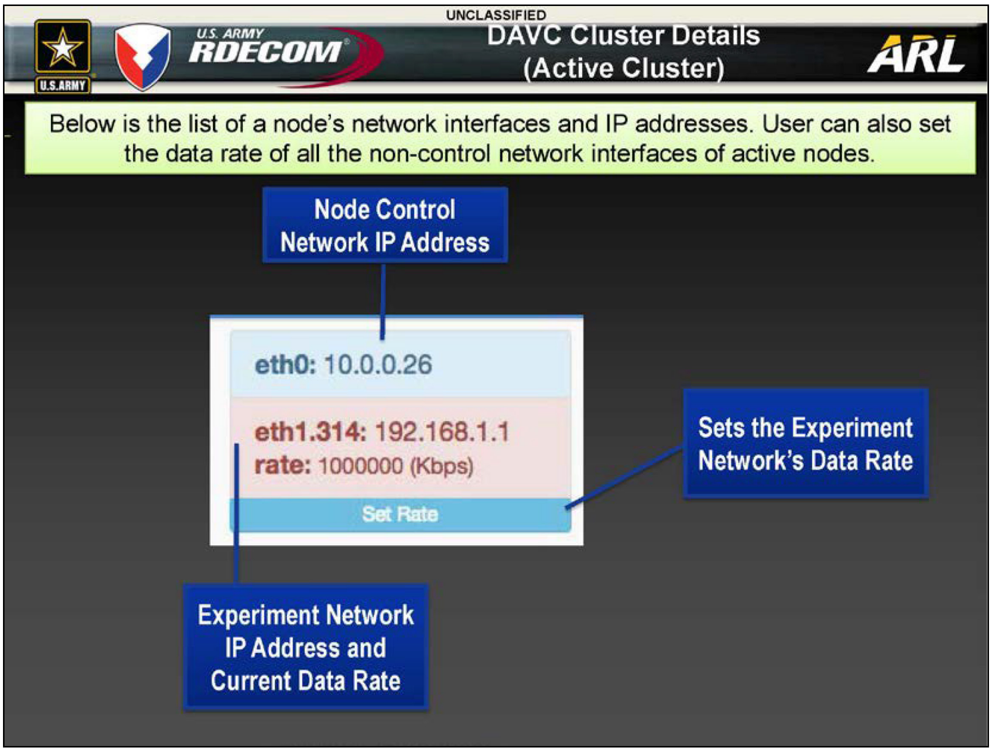

Figure D-83: DAVC Cluster Details: (Active Cluster).

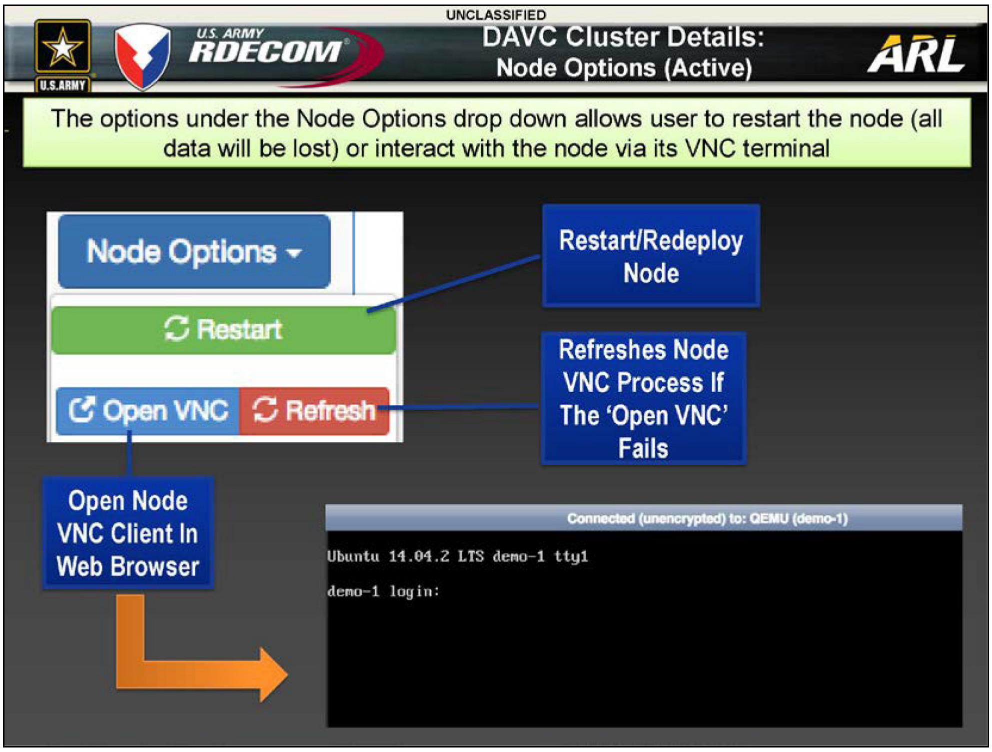

Figure D-84: DAVC Cluster Instantiation: Node Options (Active).

D.3.4 DAVC Virtual Hard Disk Management

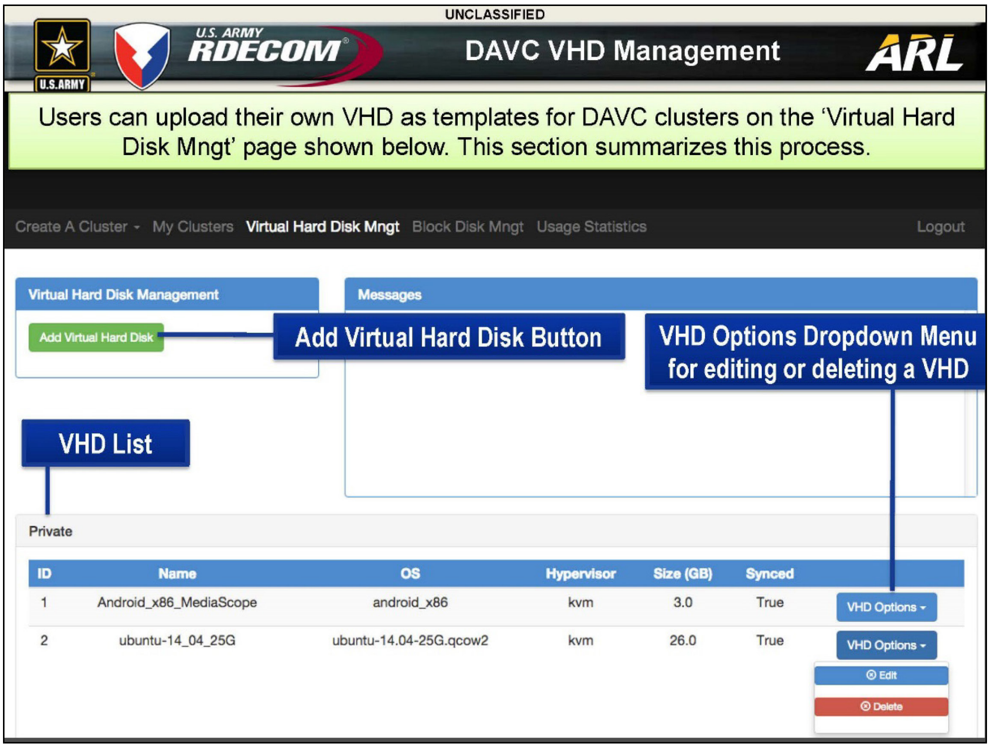

Figure D-85: DAVC VHD Management.

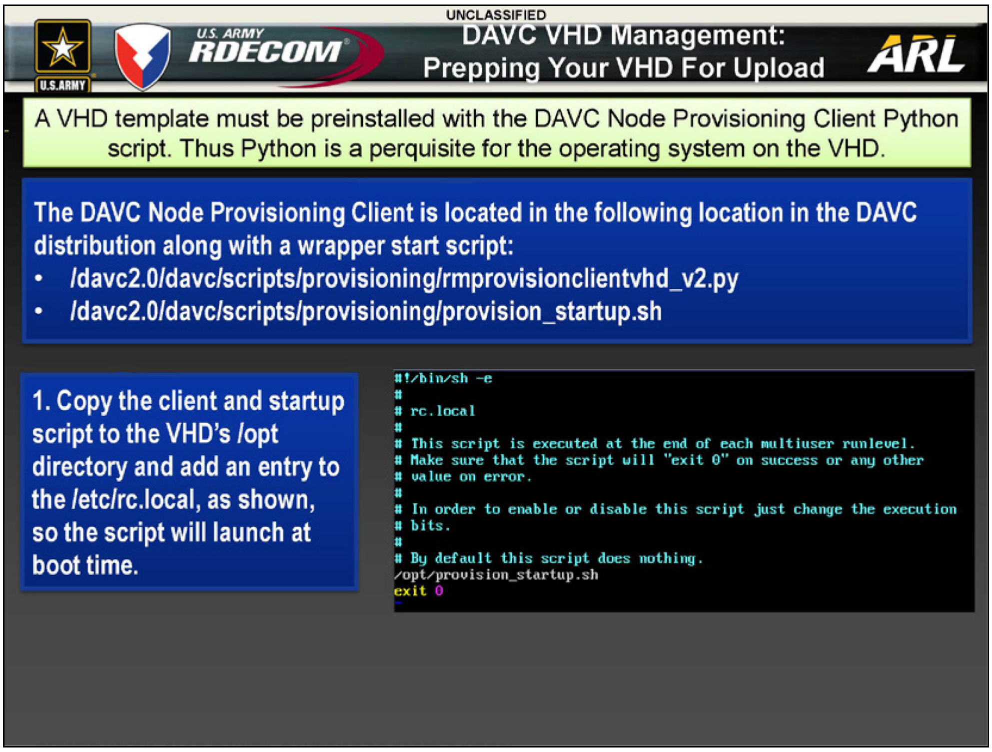

Figure D-86: DAVC VHD Management: Prepping Your VHD for Upload.

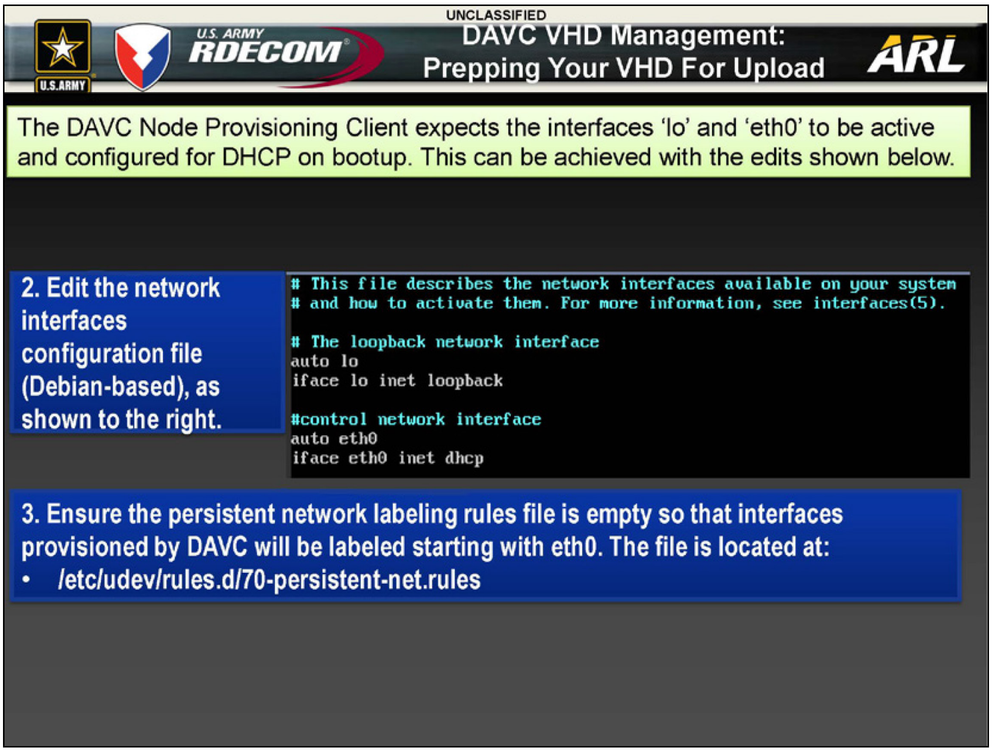

Figure D-87: DAVC VHD Management: Prepping Your VHD for Upload.

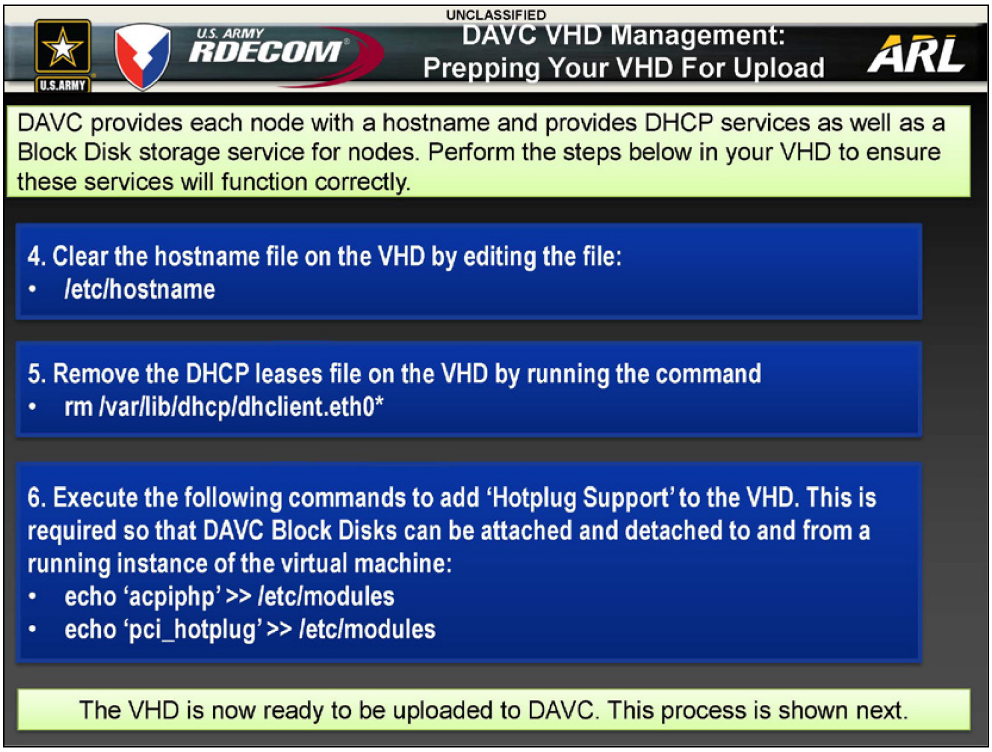

Figure D-88: DAVC VHD Management: Prepping Your VHD for Upload.

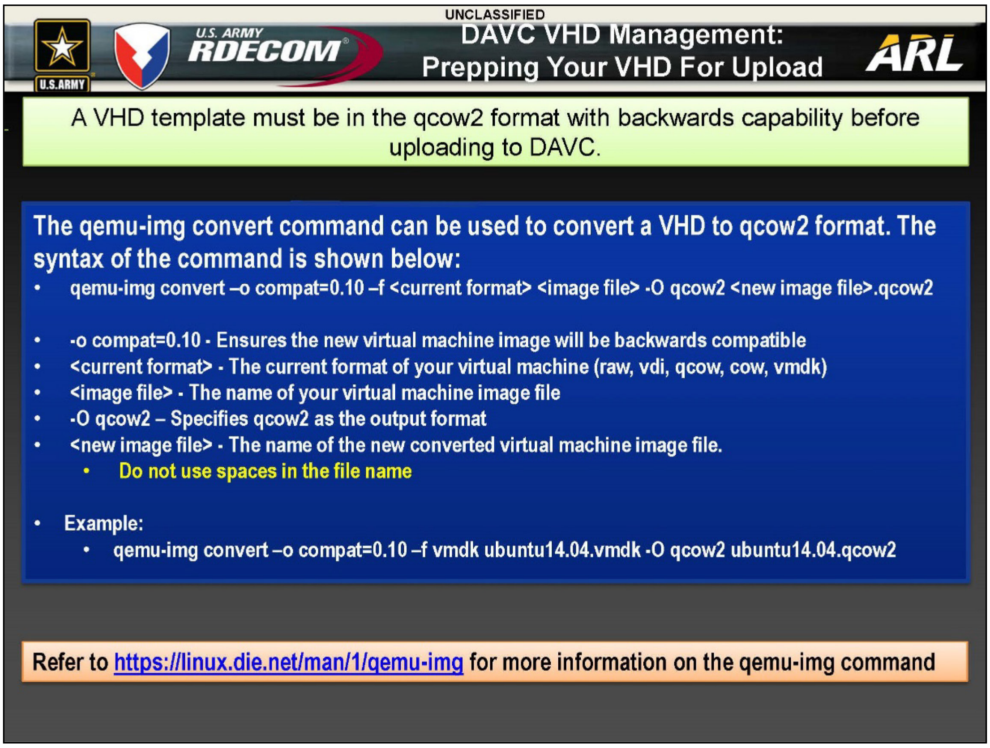

Figure D-89: DAVC VHD Management: Prepping Your VHD for Upload.

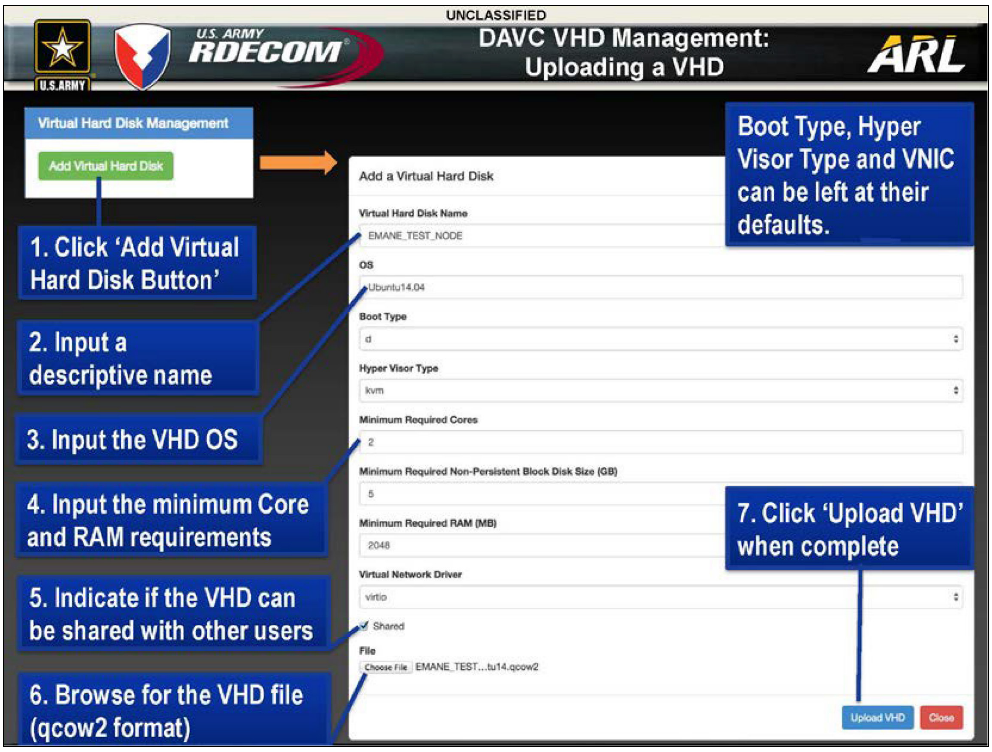

Figure D-90: DAVC VHD Management: Uploading a VHD.

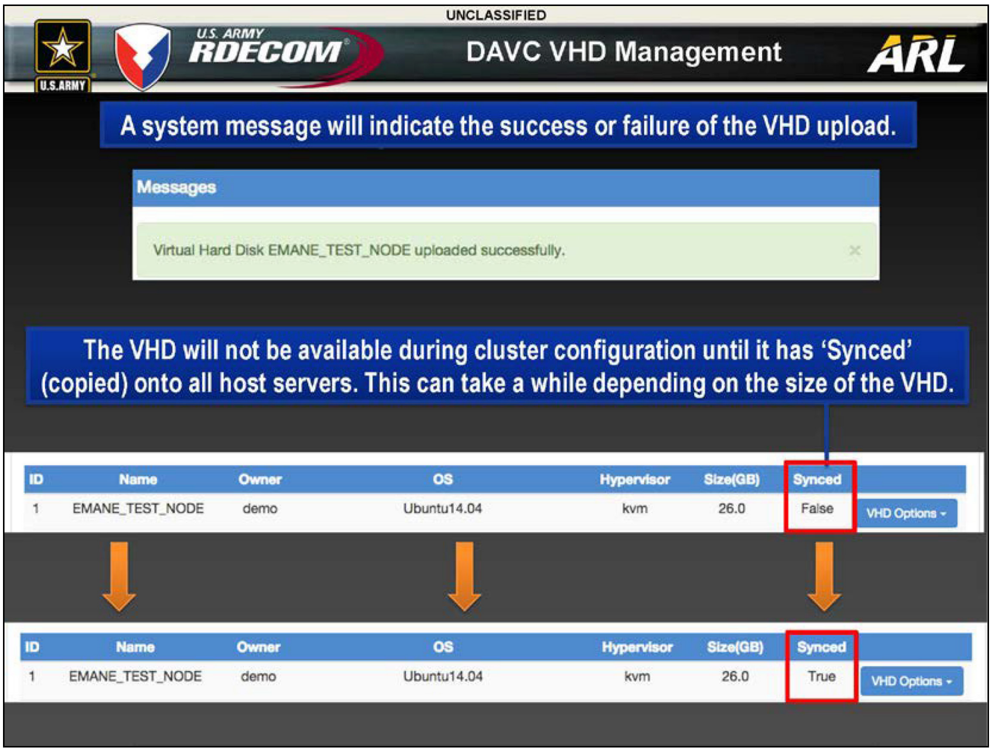

Figure D-91: DAVC VHD Management.

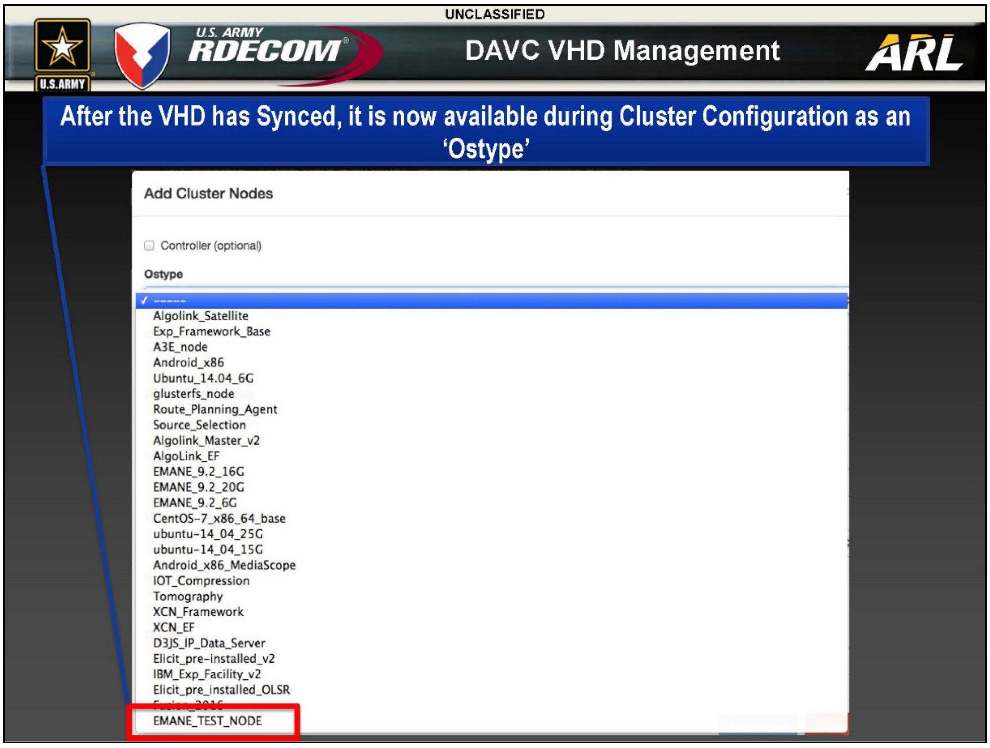

Figure D-92: DAVC VHD Management.

D.3.5 DAVC Block Disk/Persistent Storage Management

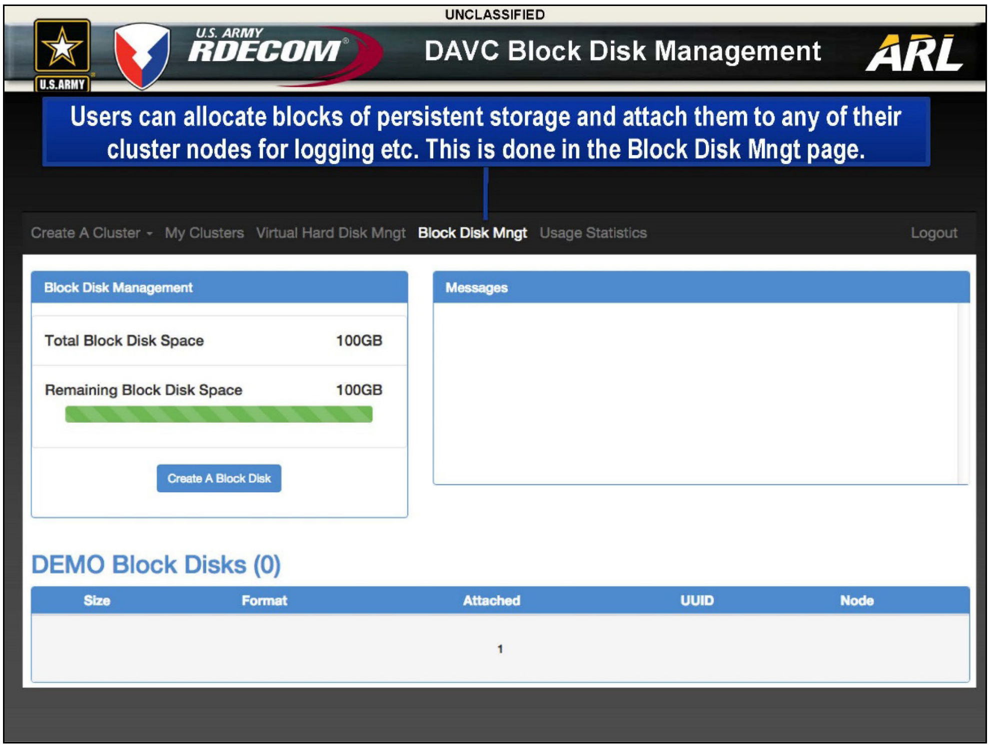

Figure D-93: DAVC Block Disk Management.

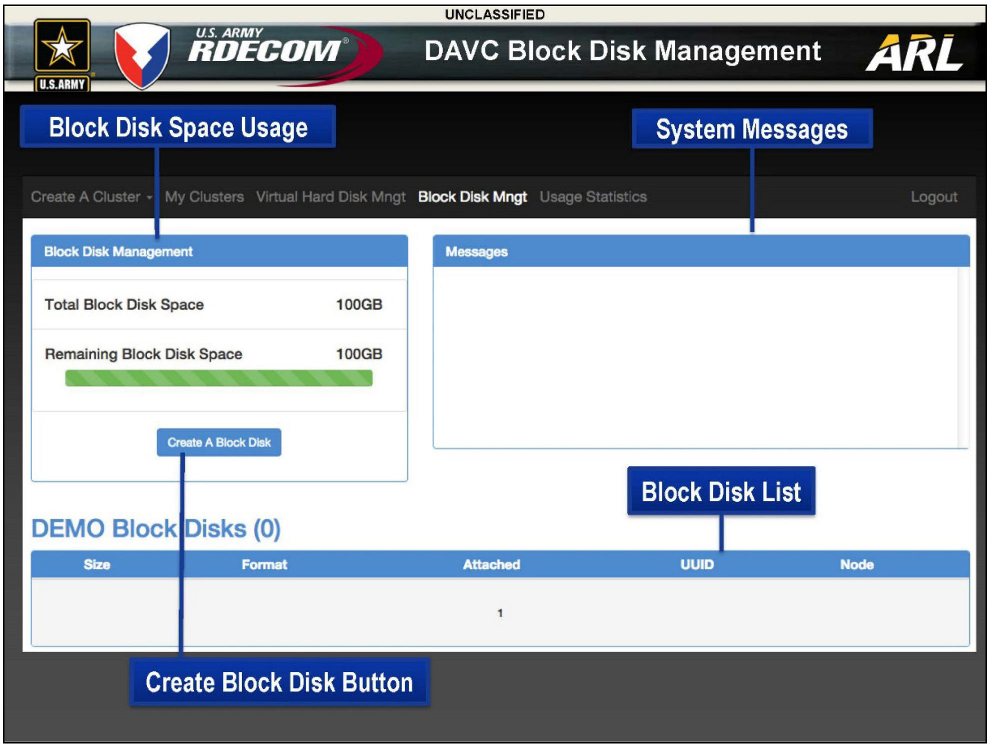

Figure D-94: DAVC Block Disk Management.

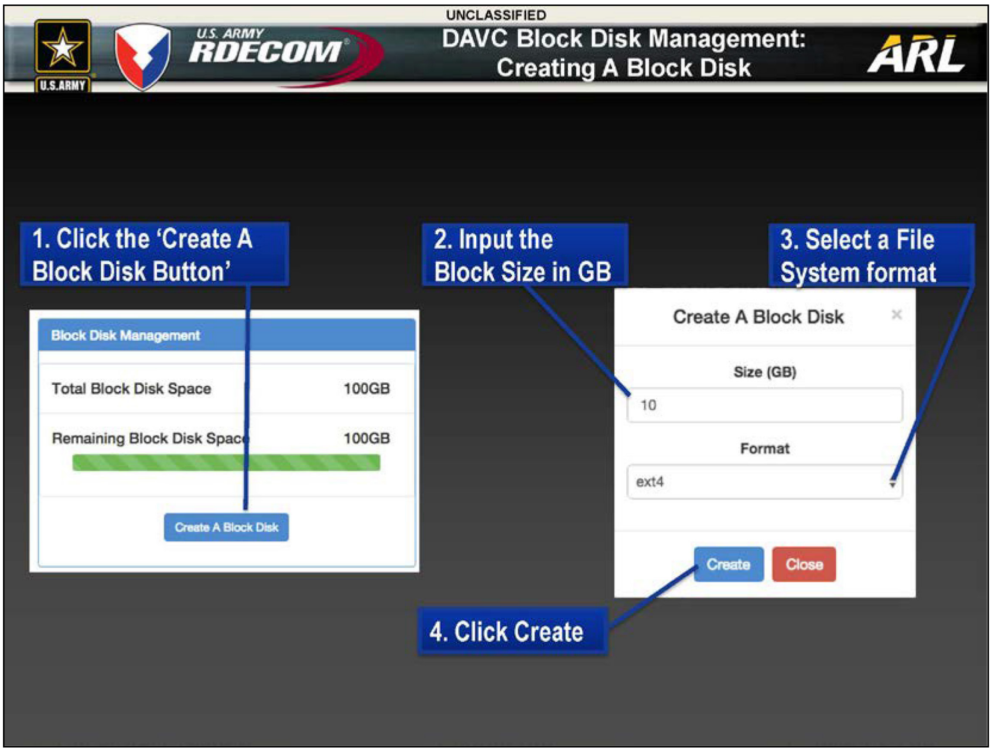

Figure D-95: DAVC Block Disk Management: Creating a Block Disk.

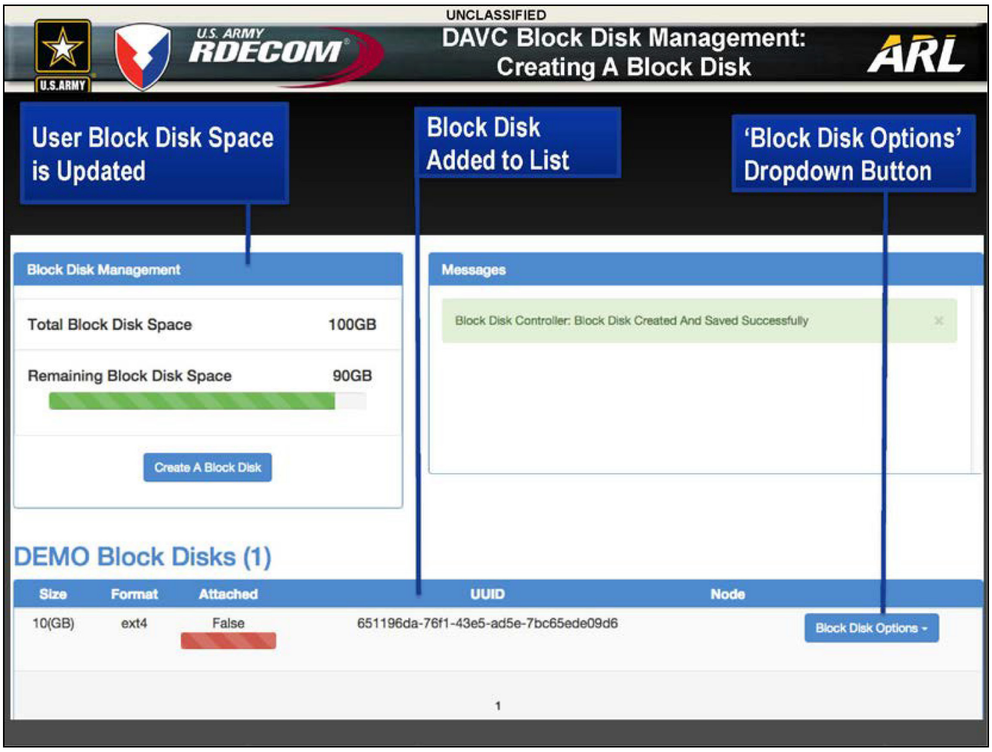

Figure D-96: DAVC Block Disk Management: Creating a Block Disk.

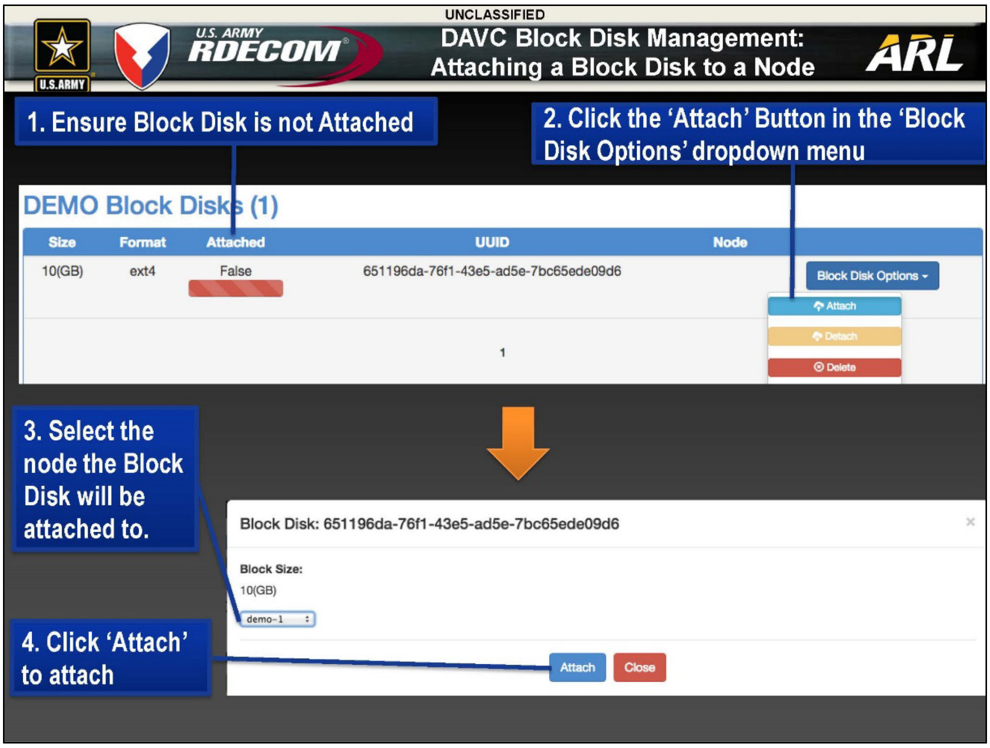

Figure D-97: DAVC Block Disk Management: Attaching a Block Disk to a Node.

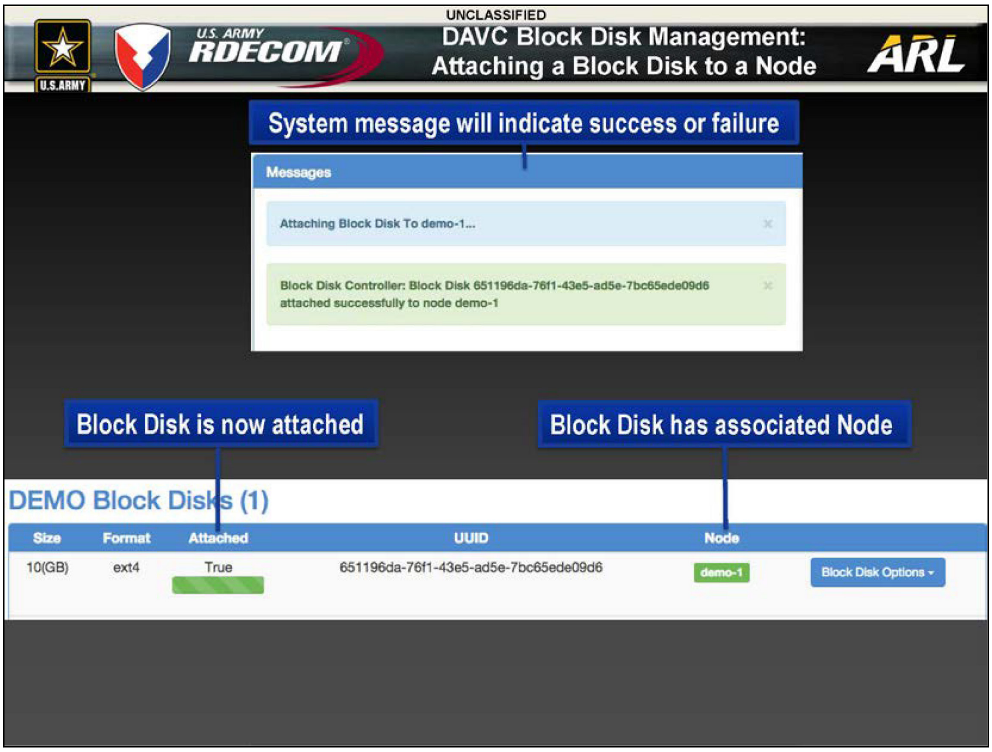

Figure D-98: DAVC Block Disk Management: Attaching a Block Disk to a Node.

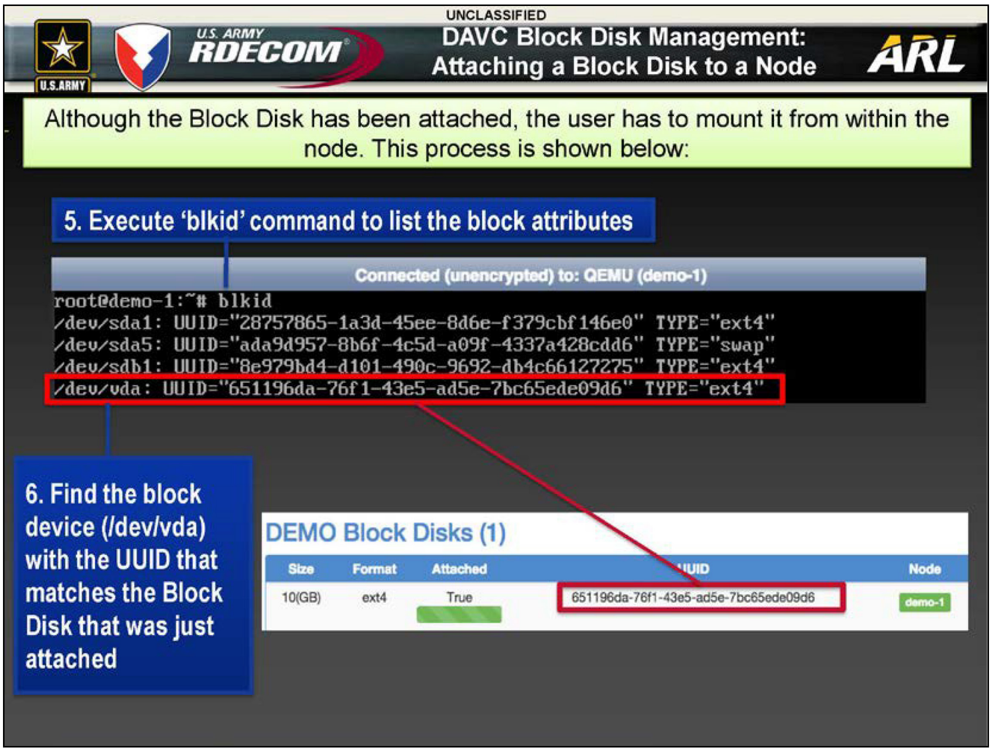

Figure D-99: DAVC Block Disk Management: Attaching a Block Disk to a Node.

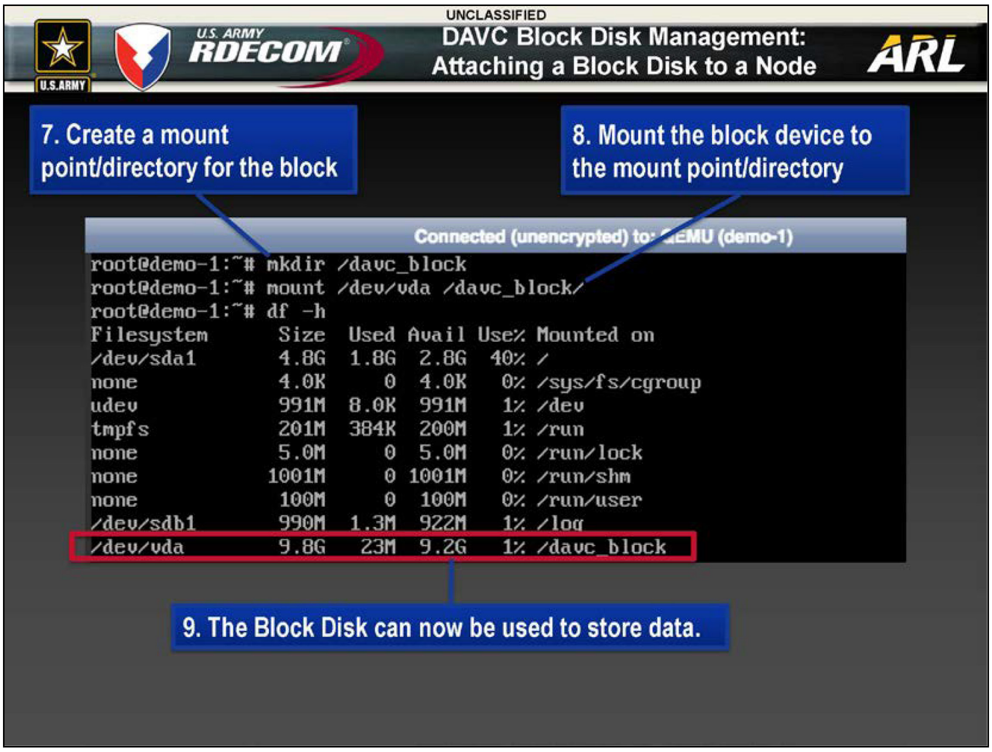

Figure D-100: DAVC Block Disk Management: Attaching a Block Disk to a Node.

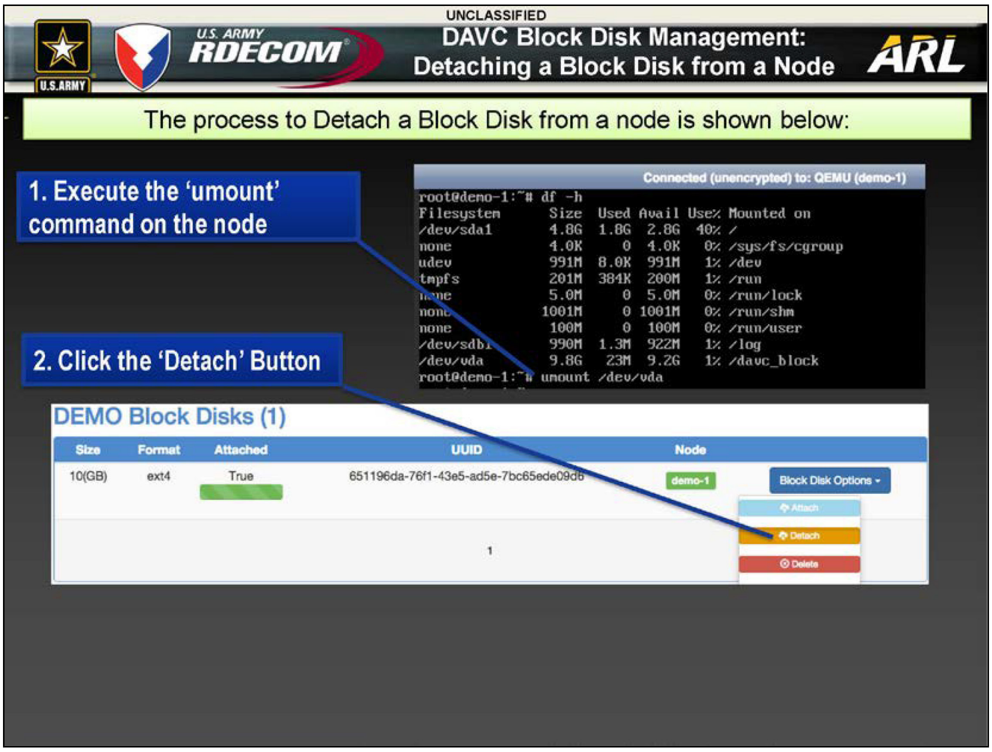

Figure D-101: DAVC Block Disk Management: Detaching a Block Disk from a Node.

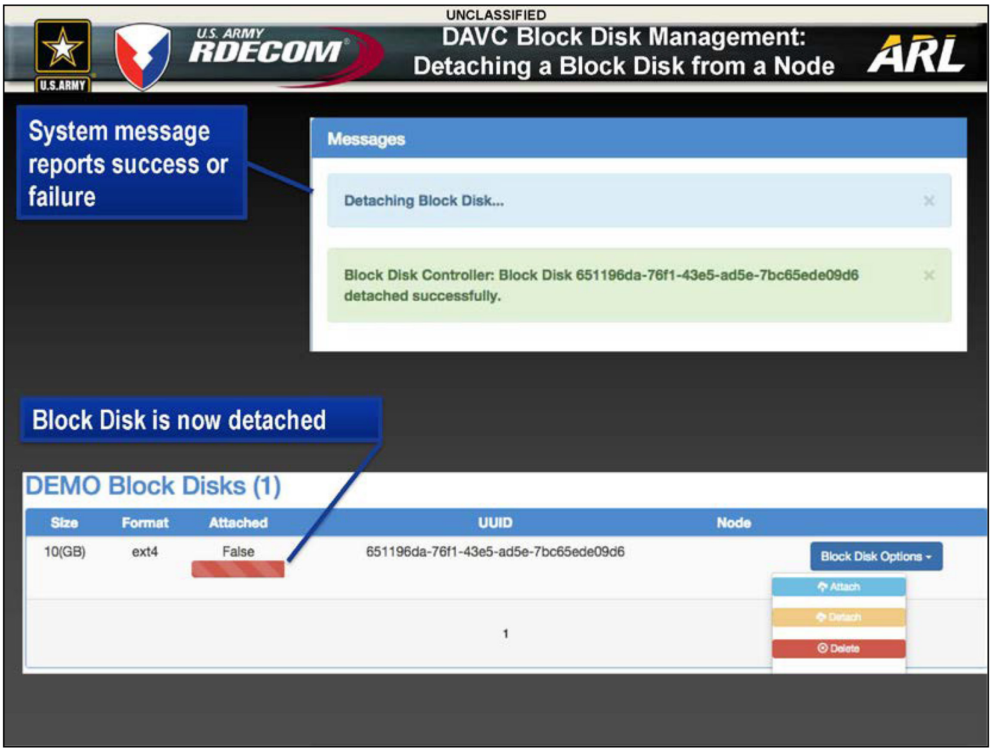

Figure D-102: DAVC Block Disk Management: Detaching a Block Disk from a Node.

D.3.6 Creating a New Virtual Hard Disk from a Cluster Node

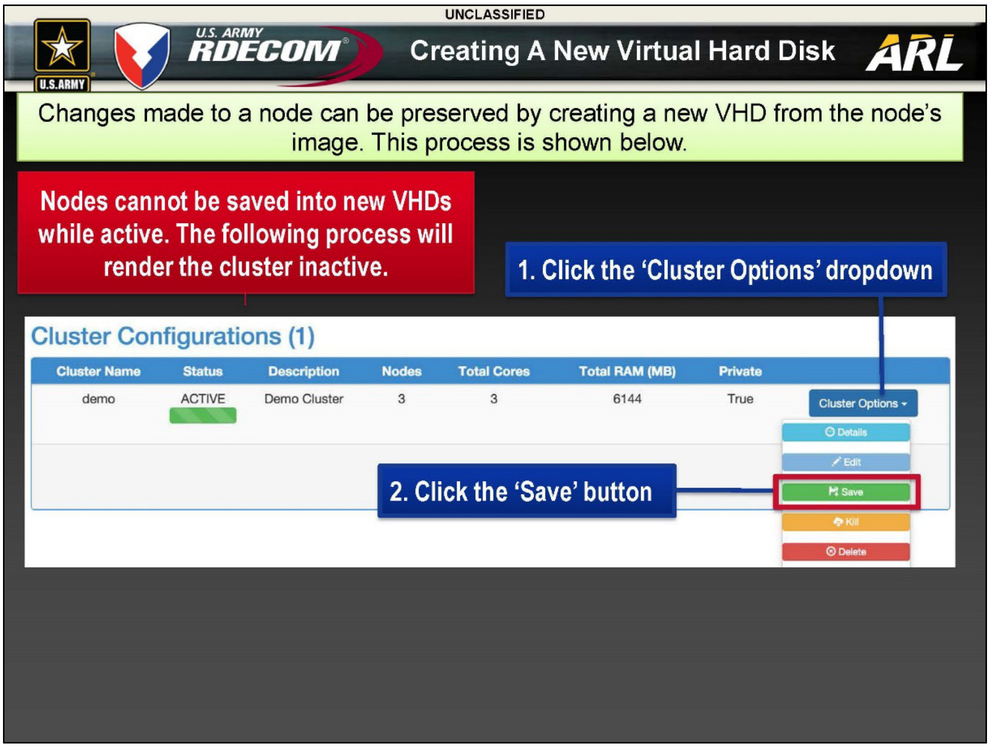

Figure D-103: Creating a New Virtual Hard Disk.

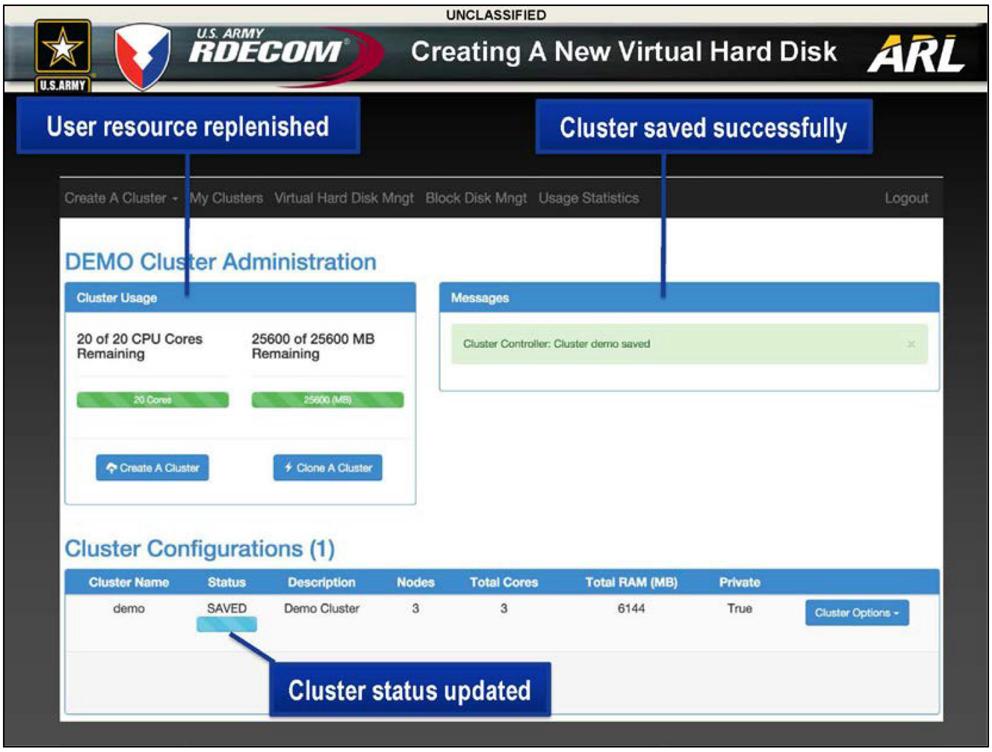

Figure D-104: Creating a New Virtual Hard Disk.

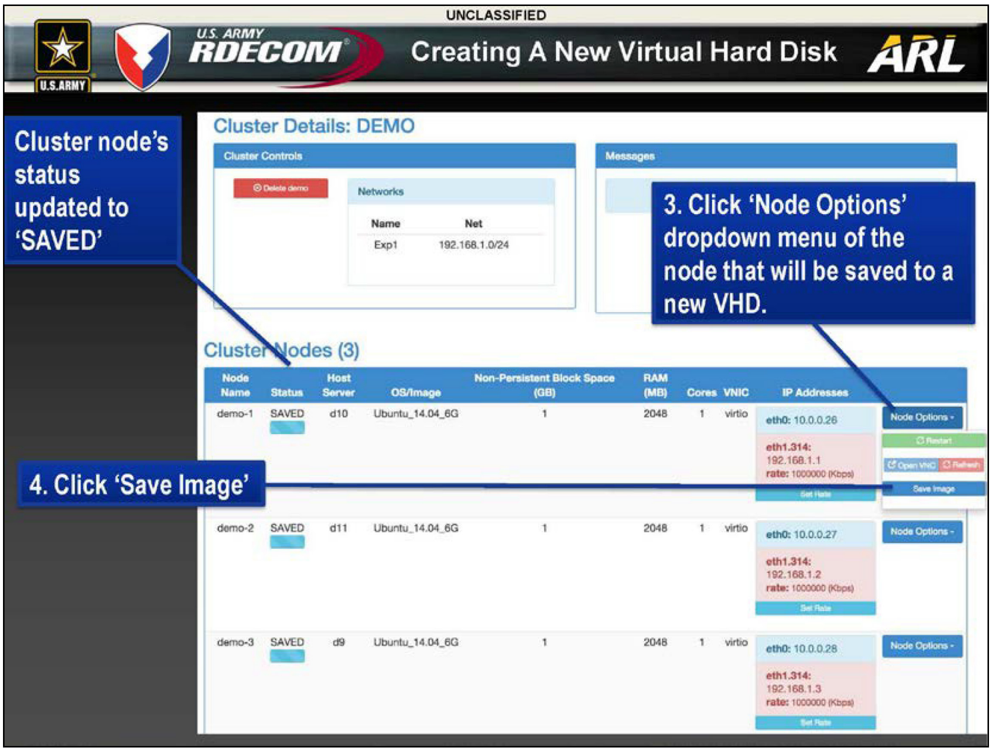

Figure D-105: Creating a New Virtual Hard Disk.

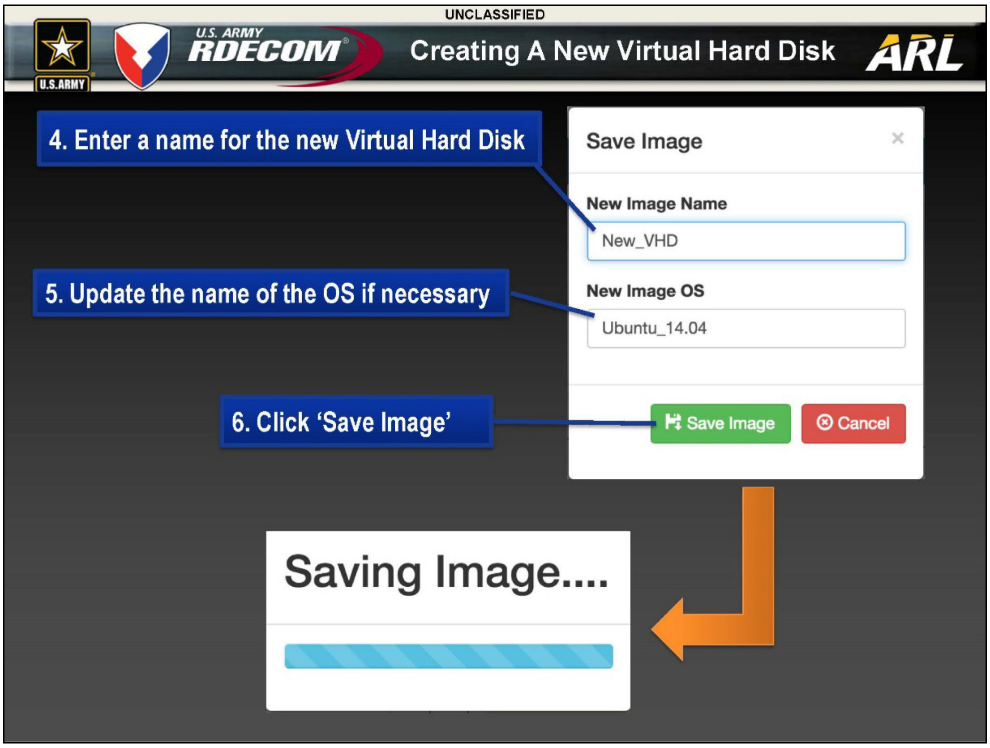

Figure D-106: Creating a New Virtual Hard Disk.

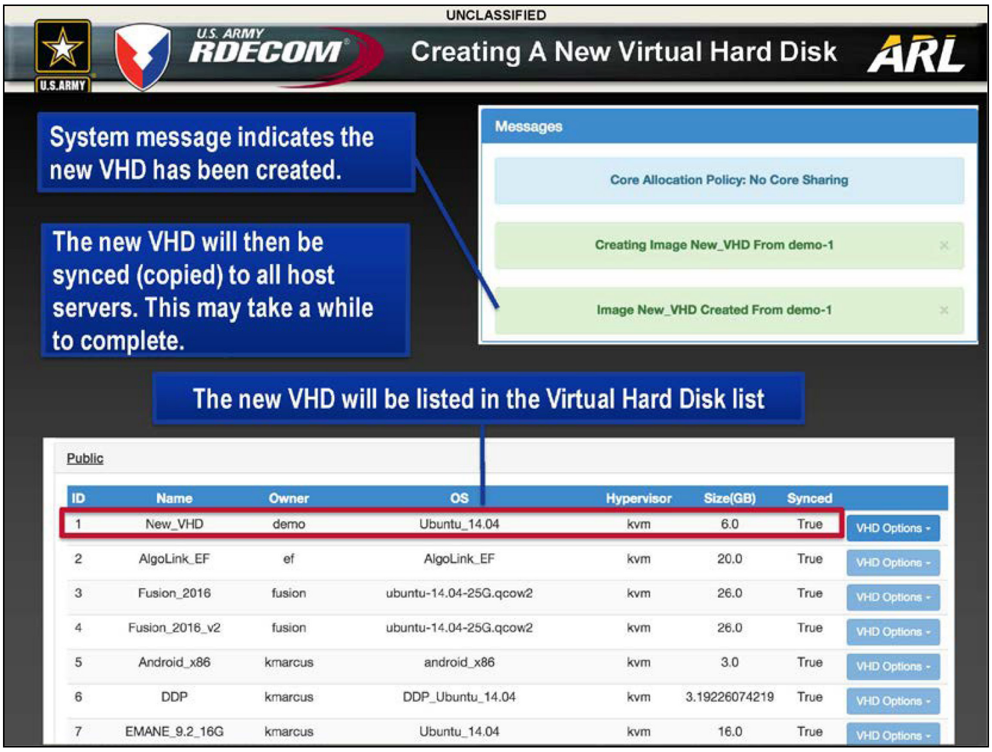

Figure D-107: Creating a New Virtual Hard Disk.

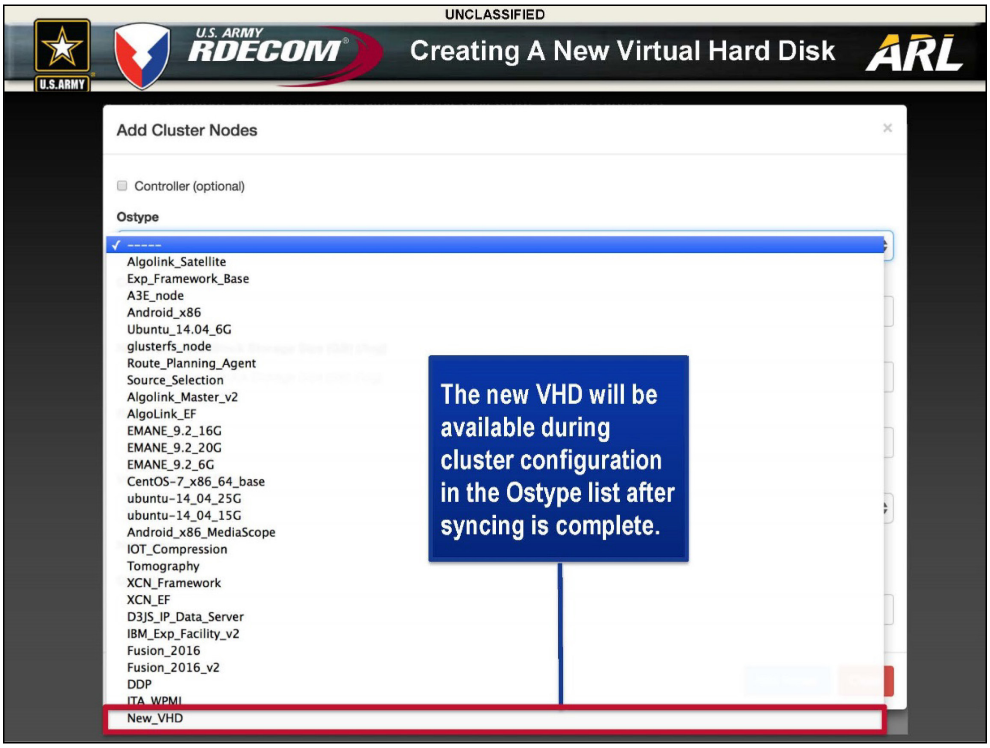

Figure D-108: Creating a New Virtual Hard Disk.

D.3.7 Conclusion

This section displayed the step-by-step instructions to perform common DAVC version 2.0 operations to access DAVC and manage DAVC clusters, nodes, virtual hard disks, and persistent block storage.

D.4 REFERENCES

[1] AdjacentLink, LLC. Extendable Mobile Ad-hoc Network Emulator (EMANE). Internet: https://github.com/adjacentlink/emane, [January 2, 2016].

[2] US Naval Research Laboratory. Multi-Generator (MGEN). Internet: http://www.nrl.navy.mil/itd/ncs/products/mgen, [January 2, 2016].

[3] Optimized Link State Routing Protocol V1 – OLSRv1. Internet: http://www.olsr.org/mediawiki/index.php/Projects, [January 2, 2016].

[4] Optimized Link State Routing Protocol V2 – OLSRv2. Internet: http://www.olsr.org/mediawiki/index.php/Projects, [January 2, 2016].

[5] US Naval Research Laboratory. Scripted Display Tools (std/std3d). Internet: http://www.nrl.navy.mil/itd/ncs/products/sdt, [January 2, 2016].