Technical white paper

Hewlett Packard Enterprise

Composable Infrastructure Integration Pack for Mesosphere DC/OS

V 2.0

Hands-on Lab Guide

Contents

Introduction

Mesosphere Enterprise DC/OS is a comprehensive platform for building, running and scaling modern enterprise applications based on Apache Mesos. It provides efficient resource isolation and sharing across distributed applications or frameworks. This software enables resource sharing in a fine-grained manner, improving cluster utilization.

When customers run Mesosphere Enterprise DC/OS on HPE, they can expect to achieve greater agility, efficiency, and resilience. HPE and Mesosphere are working together to enable the ease of deployment and management of their joint solution using OneView as the focal point of integration.

Solution Overview

This solution uses Ansible to automate and therefore dramatically speed up the deployment of a small cluster of Mesosphere DC/OS nodes on HPE infrastructure. The project was deployed in two phases. HPE C7000 BladeSystem was leveraged for this solution in phase 1; in phase 2 we used HPE Synergy. The Ansible playbook allows the user to deploy a cluster of three DC/OS master nodes, one DC/OS public agent node, and two DC/OS private agent nodes.

Solution Components

Hardware

The following hardware components were utilized:

Component	Purpose
HPE ProLiant BL460cGen8 Blades	DC/OS master and agent nodes (Phase 1)
HPE ProLian BL460c Gen9 Blade	Bootstrap node (Phase 1)
HPE SY480 Gen9	All nodes (Phase 2)

Software Stack

The following software components were utilized:

Component	**Version**
Mesosphere DC/OS	1.8.8 (Phase 1) 1.9 (Phase 2)
Ansible	2.2
Python	2.7.9
HPE OneView	3.00.05 (Phase 1) 3.00.07 (Phase 2)
HPE Insight Control server provisioning (ICsp)	7.6 (Phase 1)

Image Streamer	3.00.05 (Phase 2)
Ansible module for HPE OneView	3.1.1

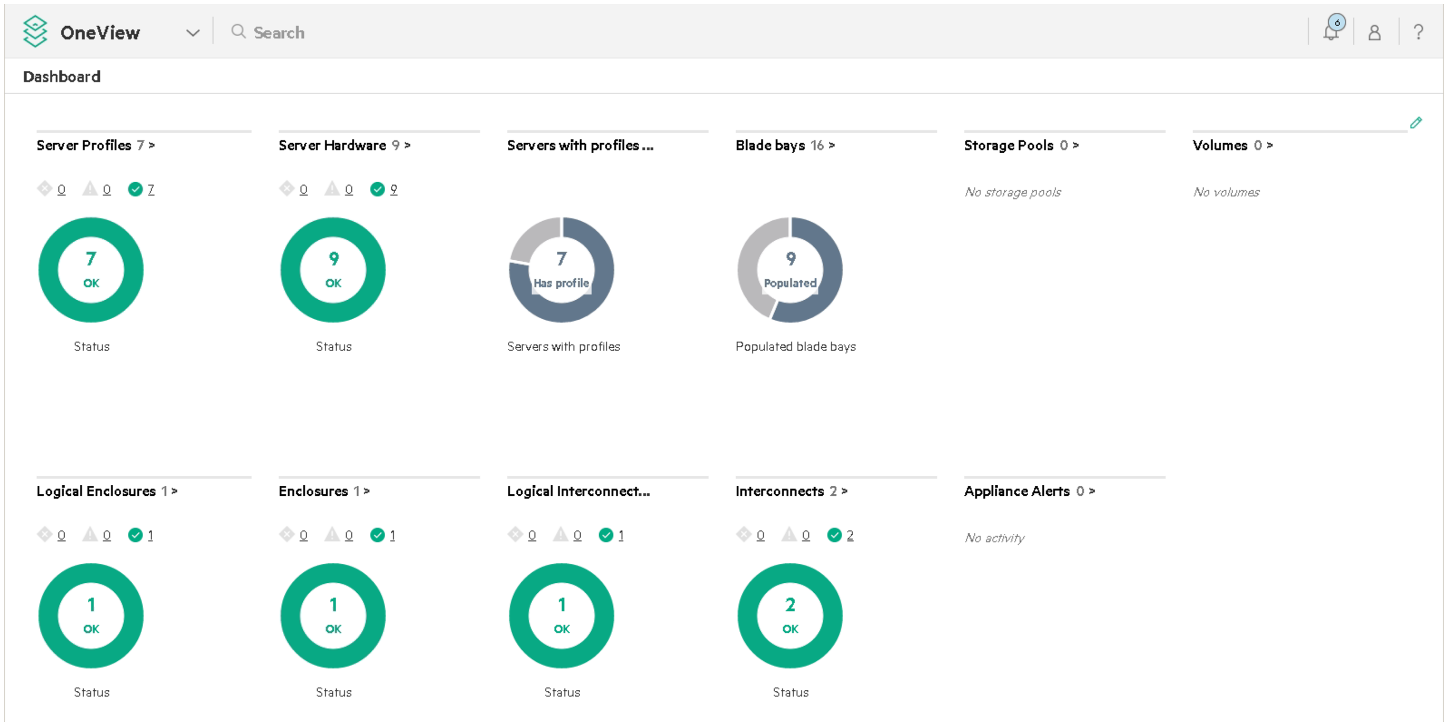
Mesosphere DC/OS

In phase 1, we used OpenSource DC/OS version 1.8.8 and 1.9.
In phase 2, we used Enterprise DC/OS version 1.9

Management Software Stack

HPE OneView

HPE OneView 3.00.05 running as a VMware appliance was used during this integration work for phase 1. Enclosure was imported, server hardware and VirtualConnect interconnect modules discovered.



For phase 2, we used HPE Composer (powered by HPE OneView) V3.00.07, and a Synergy 12000 composed of 3 frames. The frames were equipped with two Image Streamer module V3.00.05.

HPE ICsp

For the C-Class implementation (phase 1), HPE Insight Control Server Provisioning 7.6.0 running as a VMware appliance was used during this integration work. For the second phase, we did not use ICsp. A Windows based Media server was also setup to host OS distribution.

Python

The Ansible module for HPE OneView requires at least Python 2.7.9. We installed this side by side with the system Python library using the following instructions:

Step1: Install GCC

Firstly, make sure that you have gcc package installed on your system. Use following command to install gcc if you don't have it installed.

```
yum install gcc
```

Step 2: Download Python2.7

Download Python using following command from python official site. You can also download latest version in place of specified below.

```
cd /opt
wget --no-check-certificate https://www.python.org/ftp/python/2.7.13/Python-2.7.13.tgz
```

Step 3: Extract Archive and Compile

Use below set of commands to extract Python source code and compile it on your system using altinstall.

```
tar -xvf Python-2.7.13.tgz
cd Python-2.7.13
./configure
make altinstall
# make altinstall is used to prevent replacing the default python binary file
  /usr/bin/python.
```

Step 4: Check the Python Version

Check the latest version installed of python using below command

```
$ python2.7 -V
Python 2.7.13
```

HPE OneView SDK for Python

The HPE OneView SDK for Python is also a requirement. It can be installed using the following instructions:

```
git clone https://github.com/HewlettPackard/python-hpOneView.git
cd python-hpOneView
sudo python2.7 setup.py install
```

Composable Infrastructure Pack for Mesosphere DC/OS

This pack available from HPE Github bundles a set of tools used to build the entire solution

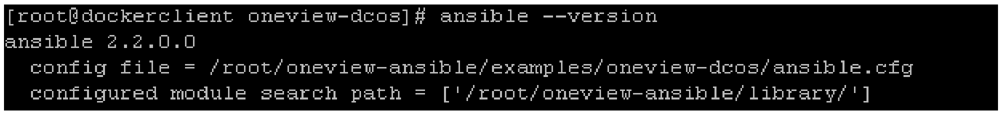
- Kickstart file for CentOS 7.2 and RedHat 7.2 (used for C-Class only)
- ICsp script for loading SSH key in target OS (used for C-Class only)

- Ansible playbooks and associated scripts for deploying the above and also the DC/OS components

You can simply clone the repo using:

```
cd ~
git clone https://github.com/HewlettPackard/dcos-hpe-oneview
```

Ansible
We used Ansible 2.2.0 on a CentOS 7 system (virtual machine), which we call the Ansible Station throughout the document.



```
[root@dockerclient oneview-dcos]# ansible --version
ansible 2.2.0.0
  config file = /root/oneview-ansible/examples/oneview-dcos/ansible.cfg
  configured module search path = ['/root/oneview-ansible/library/']
```

Ansible Module for HPE OneView/ICsp
The latest OneView module for Ansible was used. It can be installed with the following instructions:

```
cd ~
git clone https://github.com/HewlettPackard/oneview-ansible.git
export ANSIBLE_LIBRARY=~/oneview-ansible/library
export PYTHONPATH=$PYTHONPATH:$ANSIBLE_LIBRARY
```

Proof-of-concept lab environment - C7000 (Phase 1)

Compute resource configuration

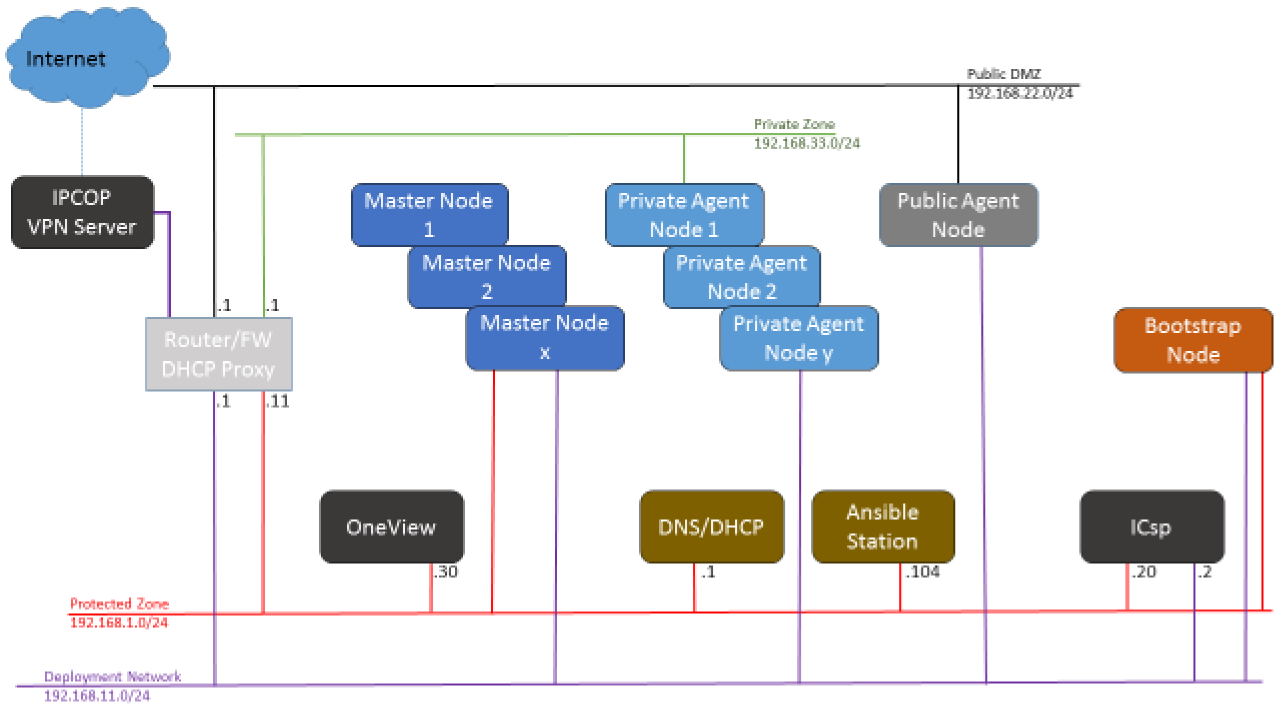
HPE assembled a proof-of-concept lab environment for DCOS which consists of 9xBL460 server (8xGen8 and 1xGen9) in a C-Class Blade C7000 enclosure. The C7000 enclosure was configured with VirtualConnect interconnect modules.

Storage configuration

THE HPE servers were configured to use local storage using the local SmartArray controller. Each server was configured with a single disk, in a RAID 0 configuration.

Network configuration

The following network configuration was implemented:



The router in the lab was configured to route between all networks. It is also configured with a DHCP Proxy Agent, so that a single DHCP Server (192.168.1.1) is able to distribute IP addresses in different scopes, one for each network. This server also provides a role of name server (DNS), with forwarder enabled to the internet. DHCP is also set to automatically register/unregister names in the DNS server.

Notes:

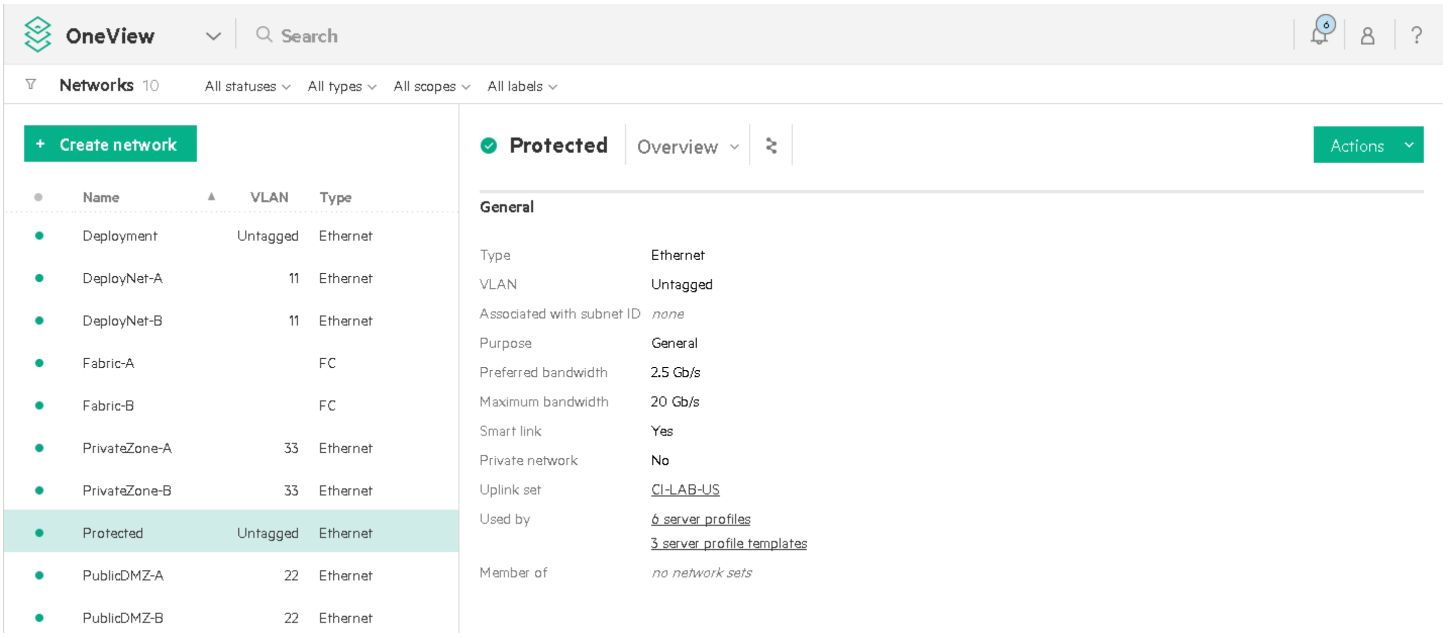
HPE servers are provisioned in the lab with DHCP. In a production environment, it is recommended to assign DHCP reservations, to make sure that IP addresses of the target DC/OS nodes will never change. This is required because DC/OS identifies machines by IP.

The network environment needs to be able to access the Internet during the provisioning of the cluster. This is the case of our lab and each network has direct access to Internet. Internet access was setup using an IPCop appliance running on VMware (http://www.ipcop.org/). IPCop was also used to provide VPN access to the lab environment.

Notes:
Configuration will need to be adapted if operations are performed behind a corporate firewall.

HPE OneView configuration

In HPE OneView we have created the networks required for the desired configuration (See diagram in chapter Network configuration)



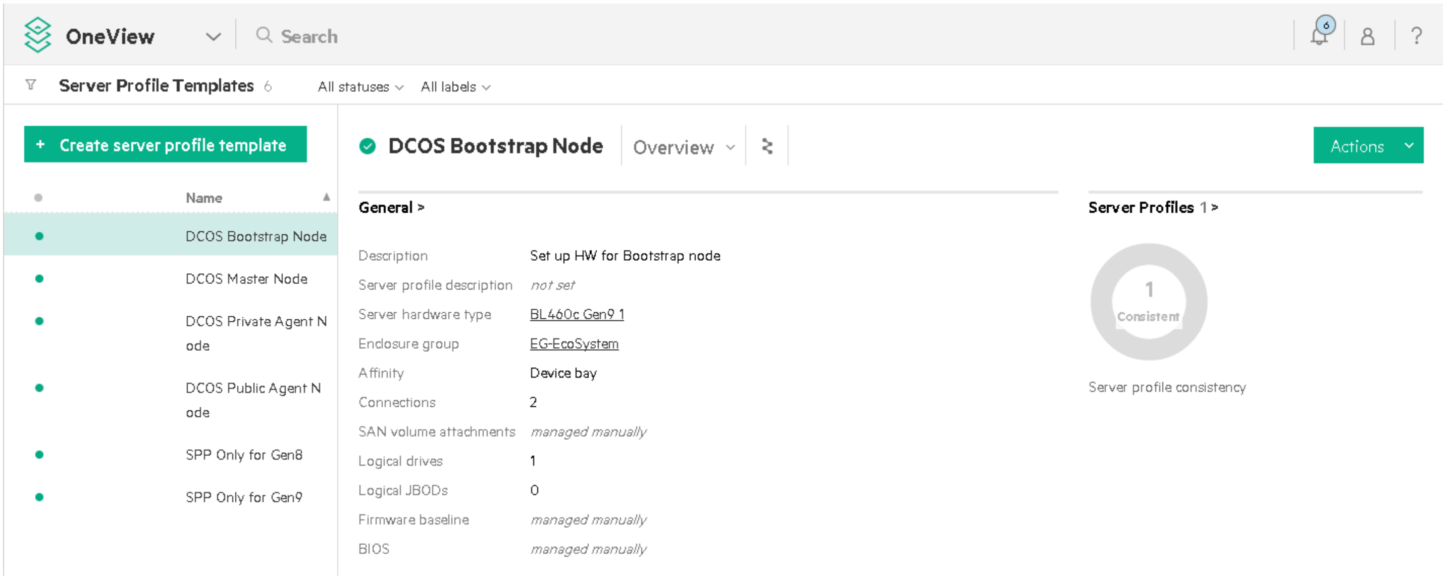
	Name	VLAN	Type
●	Deployment	Untagged	Ethernet
●	DeployNet-A	11	Ethernet
●	DeployNet-B	11	Ethernet
●	Fabric-A		FC
●	Fabric-B		FC
●	PrivateZone-A	33	Ethernet
●	PrivateZone-B	33	Ethernet
●	Protected	Untagged	Ethernet
●	PublicDMZ-A	22	Ethernet
●	PublicDMZ-B	22	Ethernet

General

Type	Ethernet
VLAN	Untagged
Associated with subnet ID	*none*
Purpose	General
Preferred bandwidth	2.5 Gb/s
Maximum bandwidth	20 Gb/s
Smart link	Yes
Private network	No
Uplink set	CI-LAB-US
Used by	6 server profiles 3 server profile templates
Member of	*no network sets*

We then configured a Server Profile Template for each type of nodes with the following details:

- Bootstrap Node: 2 network connections (Deployment and Protected Zone), BL460 Gen9
- Master Node: 2 network connections (Deployment and Protected Zone), BL460 Gen8
- Agent Node: 2 network connections (Deployment and Private Zone), BL460 Gen8
- Public Agent Node: 2 network connections (Deployment and Public DMZ), BL460 Gen8



DCOS Bootstrap Node
DCOS Master Node
DCOS Private Agent Node
DCOS Public Agent Node
SPP Only for Gen8
SPP Only for Gen9

DCOS Bootstrap Node

Description	Set up HW for Bootstrap node
Server profile description	*not set*
Server hardware type	BL460c Gen9 1
Enclosure group	EG-EcoSystem
Affinity	Device bay
Connections	2
SAN volume attachments	*managed manually*
Logical drives	1
Logical JBODs	0
Firmware baseline	*managed manually*
BIOS	*managed manually*

Notes:
Although we created those Server Profile Templates manually, chapter Full infrastructure as code shows how to create those artifacts with code

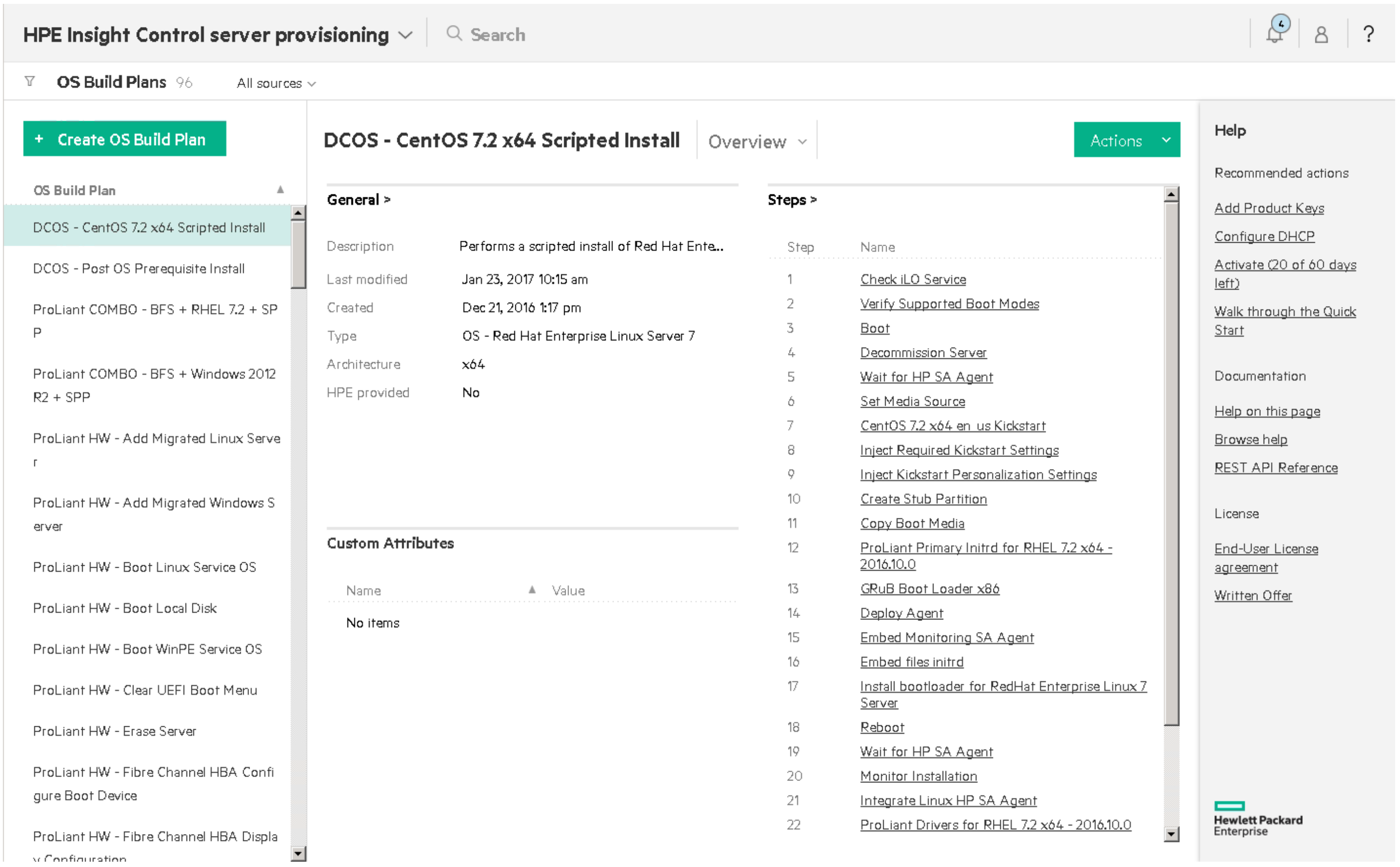
HPE ICsp Configuration

In Phase 1, we used ICsp (Insight Control Server Provisioning) to perform unattended installation of Operating Systems on target nodes. DC/OS currently supports running on CoreOS, CentOS, and RHEL. To appeal to the widest audience and simplify configuration, CentOS was chosen for this proof of concept.So we started from a standard RedHat 7.2 OS Build plan and adapted this to deploy CentOS 7.2 instead.

The following changes were implemented in the custom CentOS build plan:

1. Populate CentOS distro in ICsp Media server as /centos72-x64
2. Copy RedHat 7.2 Kickstart configuration file as CentOS 7.2 x64 en_us Kickstart
3. Edit CentOS 7.2 x64 en_us Kickstart and cut/paste the kickstart file provided as part of the Pack
4. Create a new script called DCOS - Install SSH Key and cut/paste content from script provided as part of the Pack
5. Edit Step6 of OS build plan and change Parameter to: @__OPSW-Media-LinURI@/centos72-x64
6. Edit Step7 of OS build plan and change to use CentOS 7.2 x64 en_us Kickstart created in step 2 and 3
7. Add a final step in OS build plan called, DCOS - Install SSH Key which calls the Unix script created in step 4

This OS build plan called DCOS - CentOS 7.2 x64 Scripted Install is used for the building of all our nodes in the cluster



OS Build Plan

- DCOS - CentOS 7.2 x64 Scripted Install
- DCOS - Post OS Prerequisite Install
- ProLiant COMBO - BFS + RHEL 7.2 + SPP
- ProLiant COMBO - BFS + Windows 2012 R2 + SPP
- ProLiant HW - Add Migrated Linux Server
- ProLiant HW - Add Migrated Windows Server
- ProLiant HW - Boot Linux Service OS
- ProLiant HW - Boot Local Disk
- ProLiant HW - Boot WinPE Service OS
- ProLiant HW - Clear UEFI Boot Menu
- ProLiant HW - Erase Server
- ProLiant HW - Fibre Channel HBA Configure Boot Device
- ProLiant HW - Fibre Channel HBA Display Configuration

DCOS - CentOS 7.2 x64 Scripted Install

Description	Performs a scripted install of Red Hat Ente...
Last modified	Jan 23, 2017 10:15 am
Created	Dec 21, 2016 1:17 pm
Type	OS - Red Hat Enterprise Linux Server 7
Architecture	x64
HPE provided	No

Name	Value
No items	

Step	Name
1	Check iLO Service
2	Verify Supported Boot Modes
3	Boot
4	Decommission Server
5	Wait for HP SA Agent
6	Set Media Source
7	CentOS 7.2 x64 en_us Kickstart
8	Inject Required Kickstart Settings
9	Inject Kickstart Personalization Settings
10	Create Stub Partition
11	Copy Boot Media
12	ProLiant Primary Initrd for RHEL 7.2 x64 - 2016.10.0
13	GRuB Boot Loader x86
14	Deploy Agent
15	Embed Monitoring SA Agent
16	Embed files initrd
17	Install bootloader for RedHat Enterprise Linux 7 Server
18	Reboot
19	Wait for HP SA Agent
20	Monitor Installation
21	Integrate Linux HP SA Agent
22	ProLiant Drivers for RHEL 7.2 x64 - 2016.10.0

Help

Recommended actions

- Add Product Keys
- Configure DHCP
- Activate (20 of 60 days left)
- Walk through the Quick Start

Documentation

- Help on this page
- Browse help
- REST API Reference

License

- End-User License agreement
- Written Offer

Hewlett Packard Enterprise

Notes:
Later on during the development of the solution we also validated the deployment on RHEL 7.2. The Kickstart file for RHEL7.2 is also provided in the solution. You can choose between CentOS 7.2 or RHEL7.2, by simply changing the ICsp OS build plan in the custom group_vars files

iLO Configuration

All iLO of servers involved in the DC/OS cluster should be set with a common administrative account. There are scripts to do this, leveraging the Single Sign On (SSO) from OneView and iLO of managed server-hardware (https://github.com/HewlettPackard/POSH-HPOneView/wiki/Creator-iLO)

Ansible configuration

We configured an ansible.cfg configuration file with the following parameters:

```
[defaults]
log_path = /root/log/dcos_play.log
callback_whitelist = profile_tasks
library = /root/oneview-ansible/library/
host_key_checking = False
inventory = hosts
roles_path = roles
forks = 50
```

- log_path allows us to get a log of each Playbook run in a log file
- callback_whitelist allows to get information about time spent in each play during the run.
- library points to where we installed the OneView Python library (see chapter Python library).
- inventory points to the file in the local folder, which contains the definition of our environment to build
- roles_path is the local folder where our 2 roles (dcos-node and hpe-oneview-server) are located
- forks was changed to 50 (default is 5) to allow more parallelism during task execution

The solution uses a group_var/all configuration file which has to be customized as explained below:

```
# DCOS Settings
DCOS_GENERATE_CONFIG_PATH: '/root/dcos-cache/dcos_generate_config.sh'
DCOS_CLUSTER_NAME: 'DCOS-ONEVIEW'
DCOS_BOOTSTRAP_URL: 'http://bootstrap.cilab.net'
```

It is recommended to cache the dcos_generate_config.sh file which is a big file. DCOS_GENERATE_CONFIG_PATH sets the location of the cached file on the Ansible Station. DCOS_CLUSTER_NAME sets the DC/OS Cluster name. The Boostrap URL is used by all nodes to install DC/OS

Notes:

This allows to easily change the version of the DC/OS software to use for the installation.

The following variables apply to ICsp:

```
icsp: 192.168.1.20 # ICsp appliance IP Address
icsp_username: Administrator # ICsp user name
icsp_password: <YourPasswordGoesHere> # ICsp password
osbp_custom_attributes:
  - SSH_CERT: "{{ lookup('file', '~/.ssh/root_ansible.pub') }}"
```

Notes:

The SSH_CERT points to the public SSH key to be used by Ansible.

The next section set username and password for iLOs of servers in the environment:

```
#iLO Credentials
server_username: demopaq  # iLO credentials for the target server that will be registered in ICsp
server_password: <YourPasswordGoesHere> #iLO password
```

Next, we define the DNS domain name and the DNS server IP address:

```
# Network Settings
domain_name: "cilab.net"
dns_server: 192.168.1.1
```

Finally we define the properties of the 2 interfaces, which shall be set to use DHCP:

```
network_config:
  hostname: "{{ inventory_hostname }}"
  domain: "{{ domain_name }}"
  interfaces:
  - macAddress: "{{ server_profile.connections[0].mac }}"
    enabled: true
    dhcpv4: true
    ipv6Autoconfig:
    vlanid: -1
  - macAddress: "{{ server_profile.connections[1].mac }}"
    enabled: true
    dhcpv4: true
    ipv6Autoconfig: false
  virtualInterfaces:
```

In addition to the group_vars/all, we have servers specific configuration files to store the name of the OneView server profile to use, the ICsp OS Build plan to execute and the role of each server group. We have a file for each type of server:

- group_vars/dcos-bootstrap

  ```
  # OneView Settings
  ov_template: 'DCOS Bootstrap Node'
  os_build_plan: 'DCOS - RHEL 7.2 x64 Scripted Install'
  #os_build_plan: 'DCOS - CentOS 7.2 x64 Scripted Install'
  ```

- group_vars/dcos-masters

  ```
  # OneView Settings
  ov_template: 'DCOS Master Node'
  os_build_plan: 'DCOS - RHEL 7.2 x64 Scripted Install'
  #os_build_plan: 'DCOS - CentOS 7.2 x64 Scripted Install'
  node_type: 'master'
  ```

- group_vars/dcos-private-agents

  ```
  # OneView Settings
  ov_template: 'DCOS Private Agent Node'
  os_build_plan: 'DCOS - RHEL 7.2 x64 Scripted Install'
  #os_build_plan: 'DCOS - CentOS 7.2 x64 Scripted Install'
  node_type: 'slave'
  ```

- group_vars/dcos-public-agents

  ```
  # OneView Settings
  ov_template: 'DCOS Public Agent Node'
  os_build_plan: 'DCOS - RHEL 7.2 x64 Scripted Install'
  #os_build_plan: 'DCOS - CentOS 7.2 x64 Scripted Install'
  node_type: 'slave_public'
  ```

The details about the HPE OneView appliance to use can be found in a configuration file called: config.json. This is where we set the IP address, API Version and credentials to the HPE OneView appliance.

Mesosphere DC/OS

We downloaded the DC/OS software from: http://downloads.dcos.io/dcos/stable/dcos_generate_config.sh
The file is cached on the Ansible station in a folder. By default the script looks for it in /root/dcos-cache. This is configured in group_vars/all

Using HPE Composable Infrastructure to provision DC/OS Cluster

HPE's composable infrastructure provides an unified API that uses modern REST protocols to create, aggregate, and host internal IT resources so automation tools can provision on-demand and pragmatically. Developers don't need to have a detailed understanding of the underlying physical elements. By connecting automation tools with HPE OneView, bare-metal infrastructure can be directed the same way as virtual and public cloud resources.

Use case 1 - Day 0 DC/OS Deployment

Here are the steps to provision HPE on DCOS using composable infrastructure for a configuration with:

- 1 bootstrap node (used to install all nodes)
- 3 master nodes
- 2 private agent nodes
- 1 public agent node

This is described by our Ansible inventory hosts file:

```
# There is only one bootstrap node
[dcos-bootstrap]
bootstrap

# Masters should be an odd number
[dcos-masters]
master1
master2
master3

# There are 2 types of agents, public and private
[dcos-agents:children]
dcos-public-agents
dcos-private-agents

# We start with 1 public agent
[dcos-public-agents]
agent1

# We start with 2 private agents
[dcos-private-agents]
agent2
agent3

# All nodes are either bootstrap, masters, or agents
[all-nodes:children]
dcos-bootstrap
dcos-masters
dcos-agents
```

The full provisioning of the solution is initiated by running the following command:

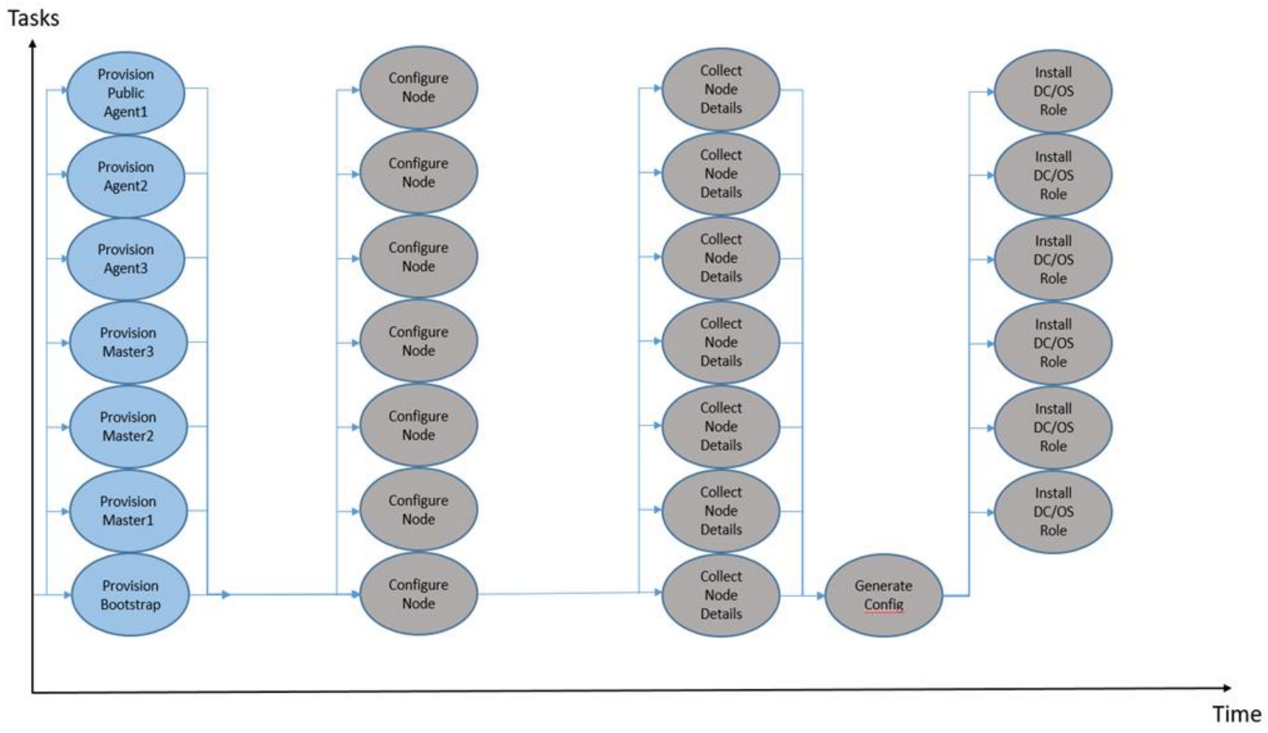
```
  ansible-playbook ov_dcos.yml -vvv -e "ansible_python_interpreter=/usr/local/bin/python2.7"
```

where ov_dcos.yml is the main playbook provided in the solution.

The playbook contains a series of plays:

```
- name: Install all Physical Nodes
  hosts: all-nodes
  vars:
 ov_osbp: "{{ os_build_plan }}"
 ov_profile: "{{ ov_template }}"
  gather_facts: no
  roles:
 - hpe-oneview-server

- name: All nodes are DC/OS Nodes
  hosts: all-nodes
  gather_facts: yes
  roles:
 - dcos-node

- name: Collect configuration of nodes
  hosts: all-nodes
  gather_facts: yes
  tasks:
  - name: Link Certificate Authorities
 # required on CentOS because DC/OS compilation is done on Ubuntu
 file: src=/etc/ssl/certs/ca-bundle.crt dest=/etc/ssl/certs/ca-certificates.crt state=link
  - include: tasks/detect-public-ip.yml

- name: Generate DC/OS Bootstrap
  hosts: dcos-bootstrap
  gather_facts: no
  tasks:
  - include: tasks/bootstrap.yml

- name: Install DC/OS Masters and Agents
  hosts: dcos-masters,dcos-agents
  gather_facts: no
  tasks:
  - include: tasks/install.yml
```

The sequencing of the task in this playbook is the following:



The blue tasks are the ones running on the Ansible machine (localhost) and using the Composable Infrastructure REST API. The grey tasks are Ansible tasks running over SSH on target machines (once provisioned with an OS)

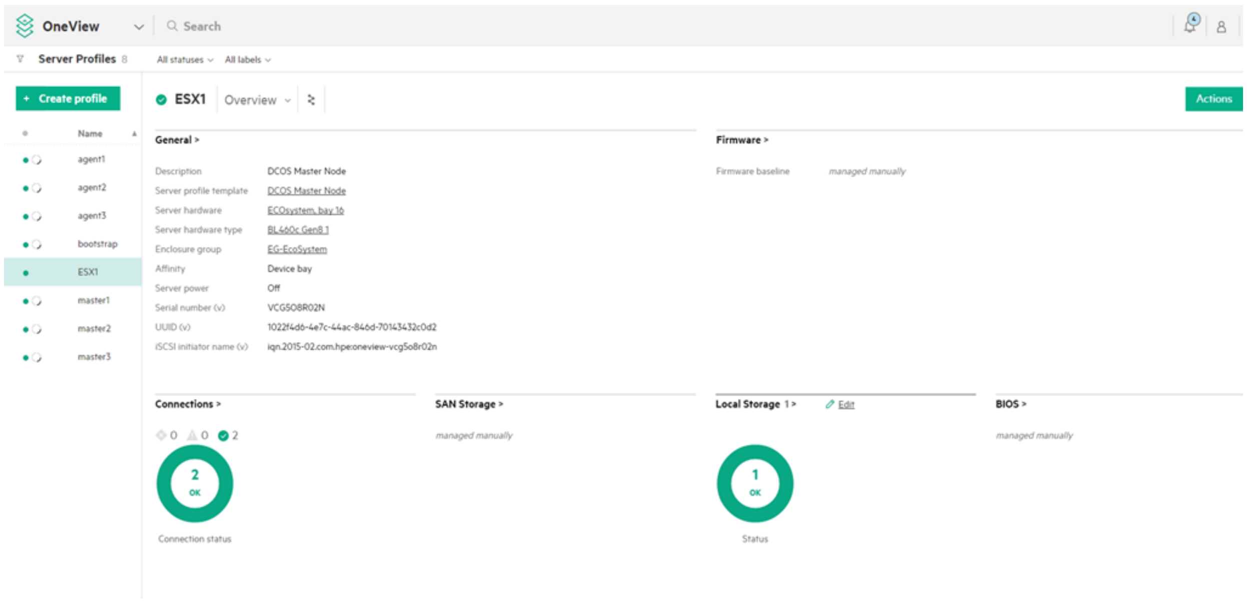
Let's look in more details each of the plays from **ov_dcos.yml**

```
- name: Install all Physical Nodes
  hosts: all-nodes
  vars:
 ov_osbp: "{{ os_build_plan }}"
 ov_profile: "{{ ov_template }}"
  gather_facts: no
  roles:
 - hpe-oneview-server
```

This task provisions all nodes in parallel. That is the bootstrap system, the masters and the agents using a **hp-oneview-server** role. The **hpe-oneview-server**, provisions a server using HPE OneView and ICsp.

The description of the **hpe-oneview-server** role is located in the folder **./roles/hpe-oneview-server**. The **file ./roles/hpe-oneview-server/tasks/main.yml** calls **./roles/hpe-oneview-server/tasks/deploy.yml**. This role will:

1. use the **group_vars/dcos-bootstrap**, **group_vars/dcos-masters**, **group_vars/dcos-private-agents** and **group_vars/dcos-public-agents** for setting group specific variables such as the name of the OneView profile templates to use and ICsp build plan to execute
2. Create the server profiles in HPE OneView from template provided and assign it to an available server-hardware in our pool of resources (this uses the HPE OneView REST API)



Server Profiles 8 | All statuses | All labels

Name
agent1
agent2
agent3
bootstrap
ESX1
master1
master2
master3

ESX1

General >

Description	DCOS Master Node
Server profile template	DCOS Master Node
Server hardware	ECOsystem, bay 16
Server hardware type	BL460c Gen8 1
Enclosure group	EG-EcoSystem
Affinity	Device bay
Server power	Off
Serial number (v)	VCG5O8R02N
UUID (v)	1022f4d6-4e7c-44ac-846d-70143432c0d2
iSCSI initiator name (v)	iqn.2015-02.com.hpe:oneview-vcg5o8r02n

Firmware >

Firmware baseline: *managed manually*

Connections >

0 0 2

2 OK

Connection status

SAN Storage >

managed manually

Local Storage 1 > Edit

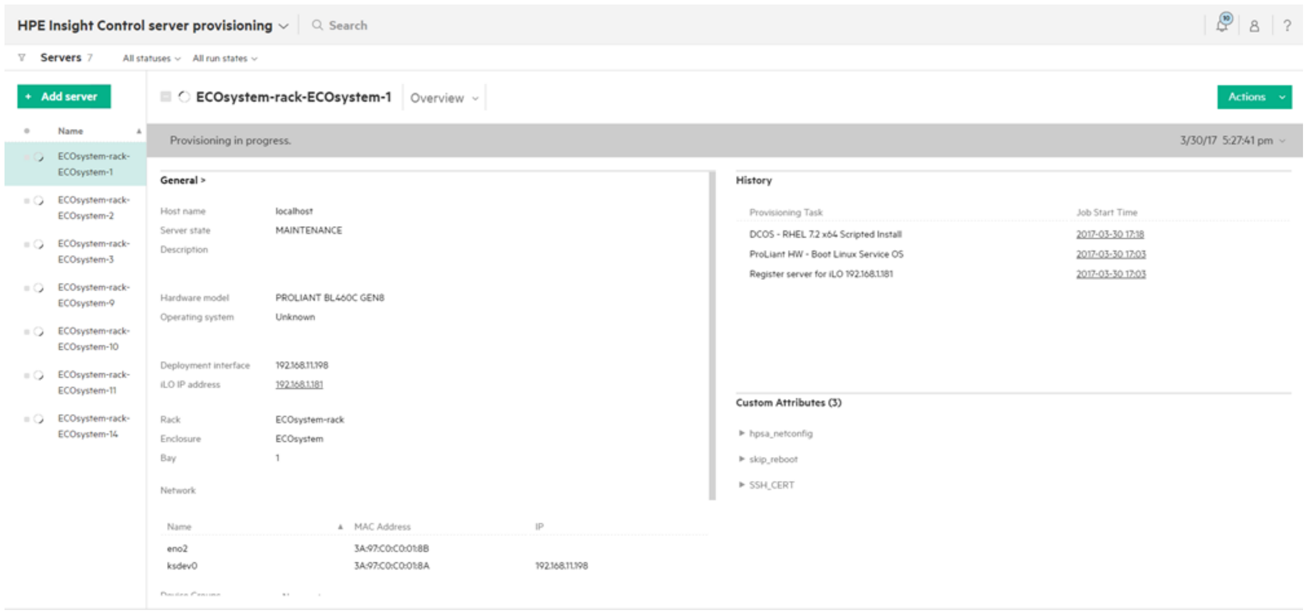
1 OK

Status

BIOS >

managed manually

3. Power the servers on
4. Register the servers in ICsp, using the iLO IP address (this uses the HPE ICsp REST API)
5. Pass the network configuration set in **group_vars/all** to ICsp (stored as ICsp custom properties). (this uses the HPE ICsp REST API)
6. Deploy the OS on these servers using ICsp, and the OS build plan provided in custom group_vars files (this uses the HPE ICsp REST API)



Servers 7 | All statuses | All run states

Name
ECOsystem-rack-ECOsystem-1
ECOsystem-rack-ECOsystem-2
ECOsystem-rack-ECOsystem-3
ECOsystem-rack-ECOsystem-9
ECOsystem-rack-ECOsystem-10
ECOsystem-rack-ECOsystem-11
ECOsystem-rack-ECOsystem-14

ECOsystem-rack-ECOsystem-1

Provisioning in progress. 3/30/17 5:27:41 pm

General >

Host name	localhost
Server state	MAINTENANCE
Description	
Hardware model	PROLIANT BL460C GEN8
Operating system	Unknown
Deployment interface	192.168.11.198
iLO IP address	192.168.1.181
Rack	ECOsystem-rack
Enclosure	ECOsystem
Bay	1

Network

Name	MAC Address	IP
eno2	3A:97:C0:C0:01:8B	
ksdev0	3A:97:C0:C0:01:8A	192.168.11.198

History

Provisioning Task	Job Start Time
DCOS - RHEL 7.2 x64 Scripted Install	2017-03-30 17:18
ProLiant HW - Boot Linux Service OS	2017-03-30 17:03
Register server for iLO 192.168.1.181	2017-03-30 17:03

Custom Attributes (3)

- hpsa_netconfig
- skip_reboot
- SSH_CERT

Notes:

The tasks in the **hpe-oneview-server** role all run on the Ansible Station, as instructed by the **delegate_to: localhost**, as illustrated below:

```
- name: Create Server Profiles
  oneview_server_profile:
     config: "{{ playbook_dir }}/../oneview_config.json"
     data:
       server_template: "{{ ov_profile }}"
       name: "{{ inventory_hostname }}"
  delegate_to: localhost
```

After this play is finished, all servers have been provisioned, and an SSH key has been installed so that the next play can continue on the target host. The next play is the following:

```
- name: All nodes are DC/OS Nodes
  hosts: all-nodes
  gather_facts: yes
  roles:
 - dcos-node
```

It applies to all-nodes and it uses another custom role called dcos-node which installs all the software prerequisites for DC/OS nodes, independent of the node types. The description of this role in located in /roles/dcos-node and the /roles/dcos-node/tasks/main.yml file describes the different steps:

1. Installs and configures docker (/roles/dcos-node/tasks/docker.yml)
2. Installs curl, bash, iputils, tar, xz, unzip, ipset, ntp, ntpdate
3. Stops the firewall
4. Starts and enables ntpd
5. Create a folder for /dcos

Steps 2, 3, 4 and 5 are described in /roles/dcos-node/tasks/dcos-deps.yml

After this task is terminated, all nodes are ready to install DC/OS.

The next step runs against all nodes, and collects public IP addresses using the task: /tasks/detect-public-ip.yml. This script, fetches the public network interface and the public IP of each node, and stores it as Ansible facts (public_iface and public_ip) for the given node:

```
- name: Collect configuration of nodes
  hosts: all-nodes
  gather_facts: yes
  tasks:
  - name: Link Certificate Authorities
 # required on CentOS because DC/OS compilation is done on Ubuntu
 file: src=/etc/ssl/certs/ca-bundle.crt dest=/etc/ssl/certs/ca-certificates.crt state=link
  - include: tasks/detect-public-ip.yml
```

The script to extract the public IP address is the following:

```
- name: Set Public Network Interface
```

```
      shell: "ifconfig | grep eno | cut --delimiter=: -f 1"
      register: if_name

    - set_fact: public_iface={{ if_name.stdout }}

    - name: Detect Public IP
      set_fact: public_ip={{ hostvars[inventory_hostname]['ansible_' +
    public_iface]['ipv4']['address'] }}

    - debug: var=public_ip
```

Once we know the IP layout of the provisioned servers we can start preparing the DC/OS installation on the bootstrap node. This done by a script called /tasks/bootstrap.yml

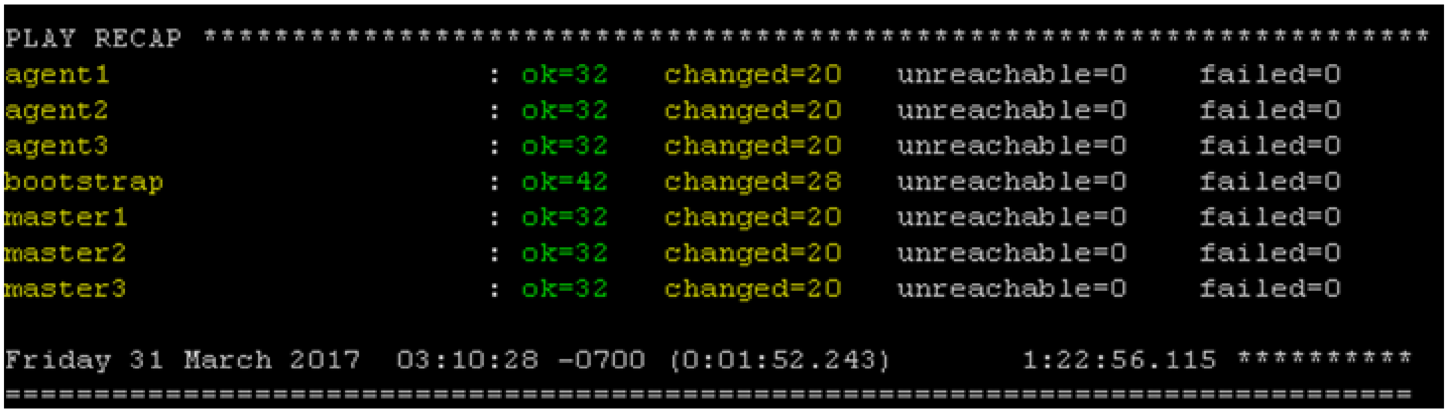
```
    - name: Generate DC/OS Bootstrap
      hosts: dcos-bootstrap
      gather_facts: no
      tasks:
      - include: tasks/bootstrap.yml
```

This is use use dcos_generate_config.sh to generate a DC/OS configuration, based on the IP layout. It also starts a nginx docker container to host the configuration. Once this is finished, the bootstrap node is ready to accept other nodes installation over the URL: http://bootstrap.cilab.net. We can initiate the last step, which installs masters and agents (in parallel)

```
    - name: Install DC/OS Masters and Agents
      hosts: dcos-masters,dcos-agents
      gather_facts: no
      tasks:
      - include: tasks/install.yml
```

The tasks /tasks/install.yml expects a variable called node_type set to either master, slave or public_slave. We set this variable in the group_var custom files.

When the playbooks terminates, we can see that it was successful (also it took about 1 hours and 22 minutes):



```
PLAY RECAP *********************************************************************
agent1                     : ok=32   changed=20   unreachable=0    failed=0
agent2                     : ok=32   changed=20   unreachable=0    failed=0
agent3                     : ok=32   changed=20   unreachable=0    failed=0
bootstrap                  : ok=42   changed=28   unreachable=0    failed=0
master1                    : ok=32   changed=20   unreachable=0    failed=0
master2                    : ok=32   changed=20   unreachable=0    failed=0
master3                    : ok=32   changed=20   unreachable=0    failed=0

Friday 31 March 2017  03:10:28 -0700 (0:01:52.243)       1:22:56.115 **********
===============================================================================
```

To validate the configuration we can run the following validation procedure provided by Mesosphere:

1. Install jq

```
wget https://github.com/stedolan/jq/releases/download/jq-1.5/jq-linux64
chmod +x ./jq-linux64
cp jq-linux64 /usr/bin/jq
```

2. Retrieve public agents IP addresses

```
curl -s http://localhost:1050/system/health/v1/nodes | jq -r '.[] | map(select(.role ==
"agent_public")) | .[].host_ip'
192.168.22.101
```

3. Retrieve private agents IP addresses

```
curl -s http://localhost:1050/system/health/v1/nodes | jq -r '.[] | map(select(.role ==
"agent")) | .[].host_ip'
192.168.33.102
192.168.33.101
```

4. Export those IP and install dig from bind-utils

```
export SLAVE_HOSTS=192.168.33.101,192.168.33.102
export PUBLIC_SLAVE_HOSTS=192.168.22.101
yum install bind-utils -y
```

5. Add test user

```
source /opt/mesosphere/environment.export
python /opt/mesosphere/active/dcos-oauth/bin/dcos_add_user.py albert@bekstil.net
source /opt/mesosphere/active/dcos-integration-test/test_env.export
cd /opt/mesosphere/active/dcos-integration-test
```

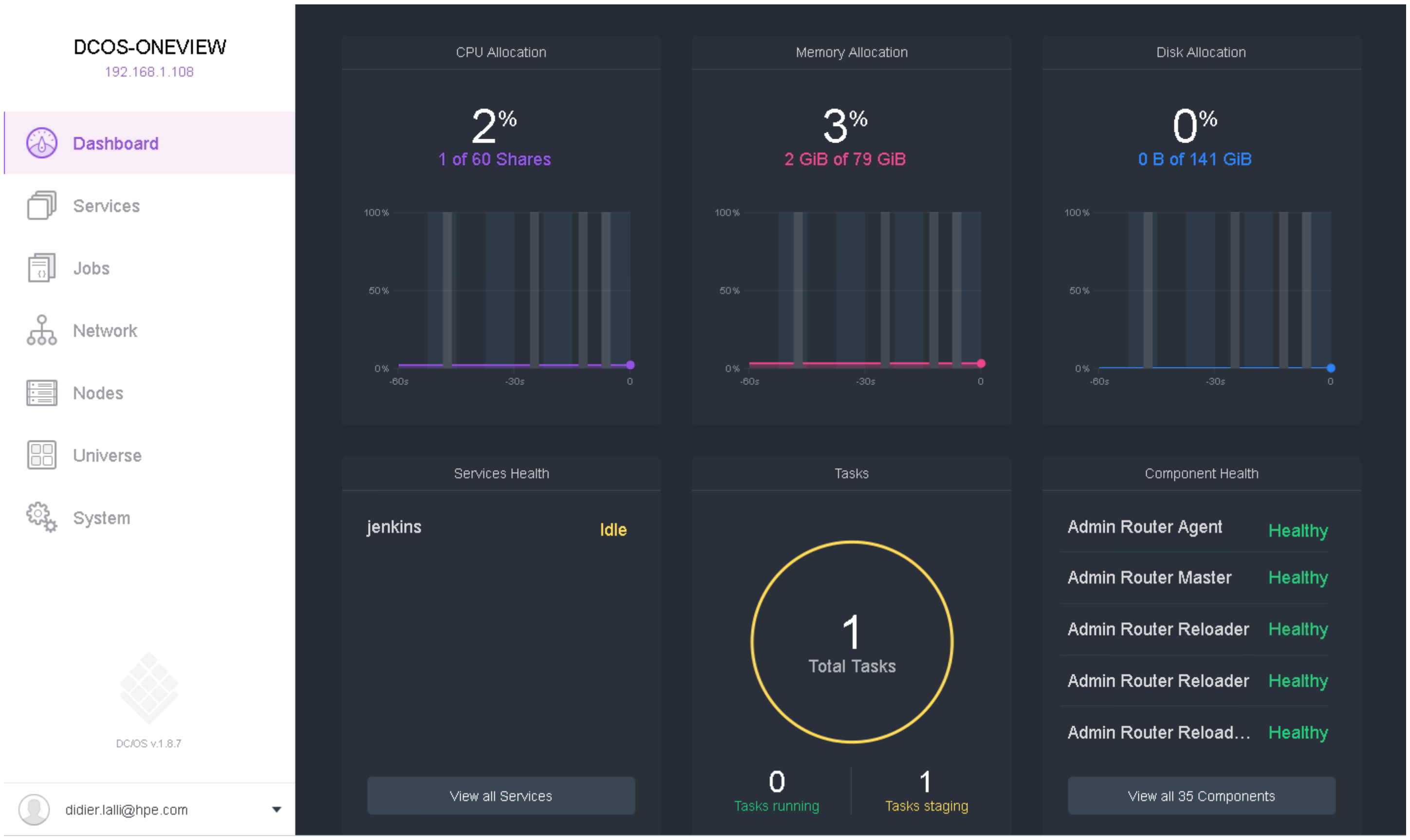
6. Run validation test

```
py.test -m 'not ccm'
```

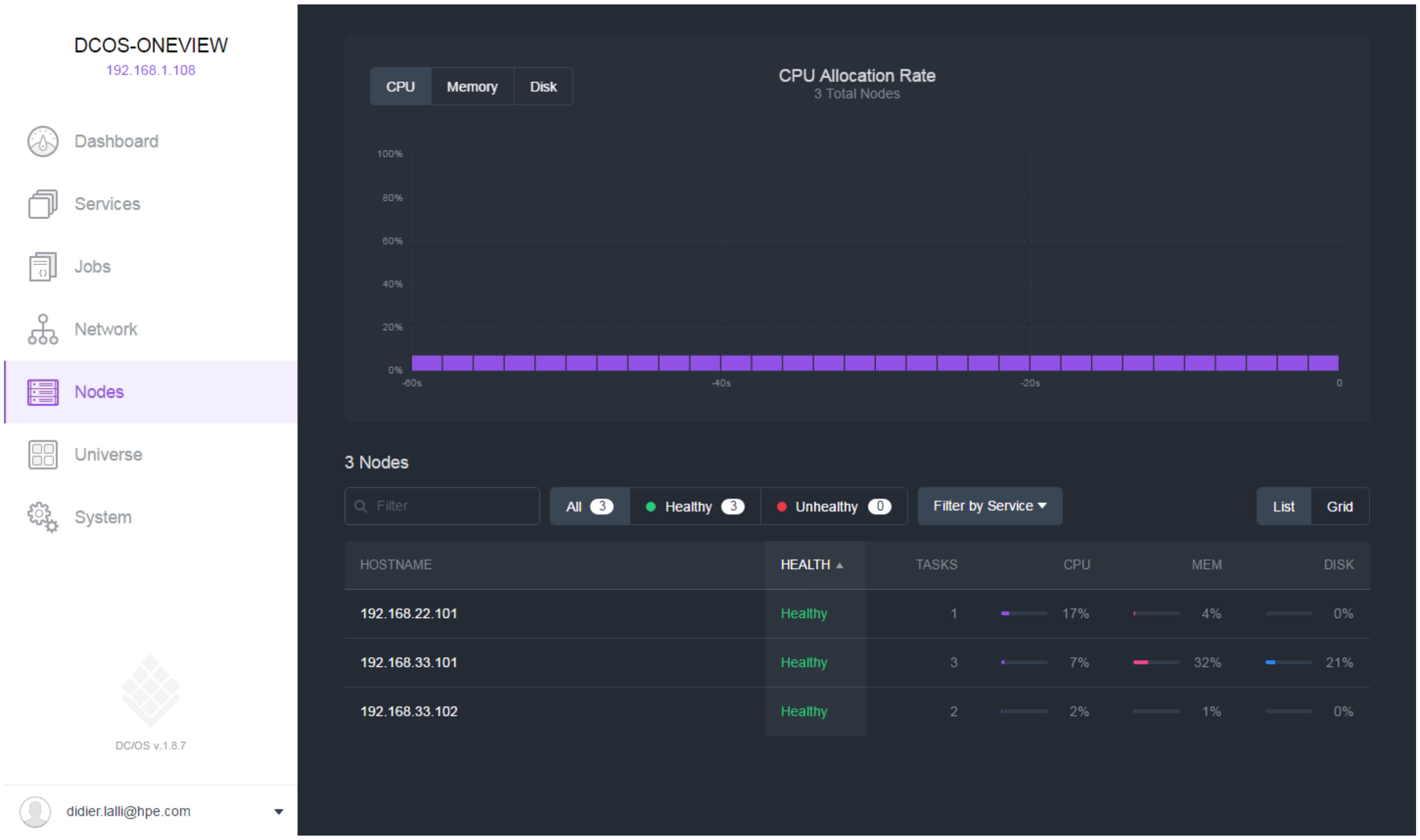
This should result in the following:

```
==================== 1 tests deselected by "-m 'not ccm'" ====================
============= 52 passed, 3 skipped, 1 deselected in 372.21 seconds =============
```

Once this is validated, we can now operate our brand new DC/OS cluster by accessing http://master1/#/dashboard/ or any of the other masters.



We can also select the Node view to validate our 3 Agents (1 public and 2 private), up and running.



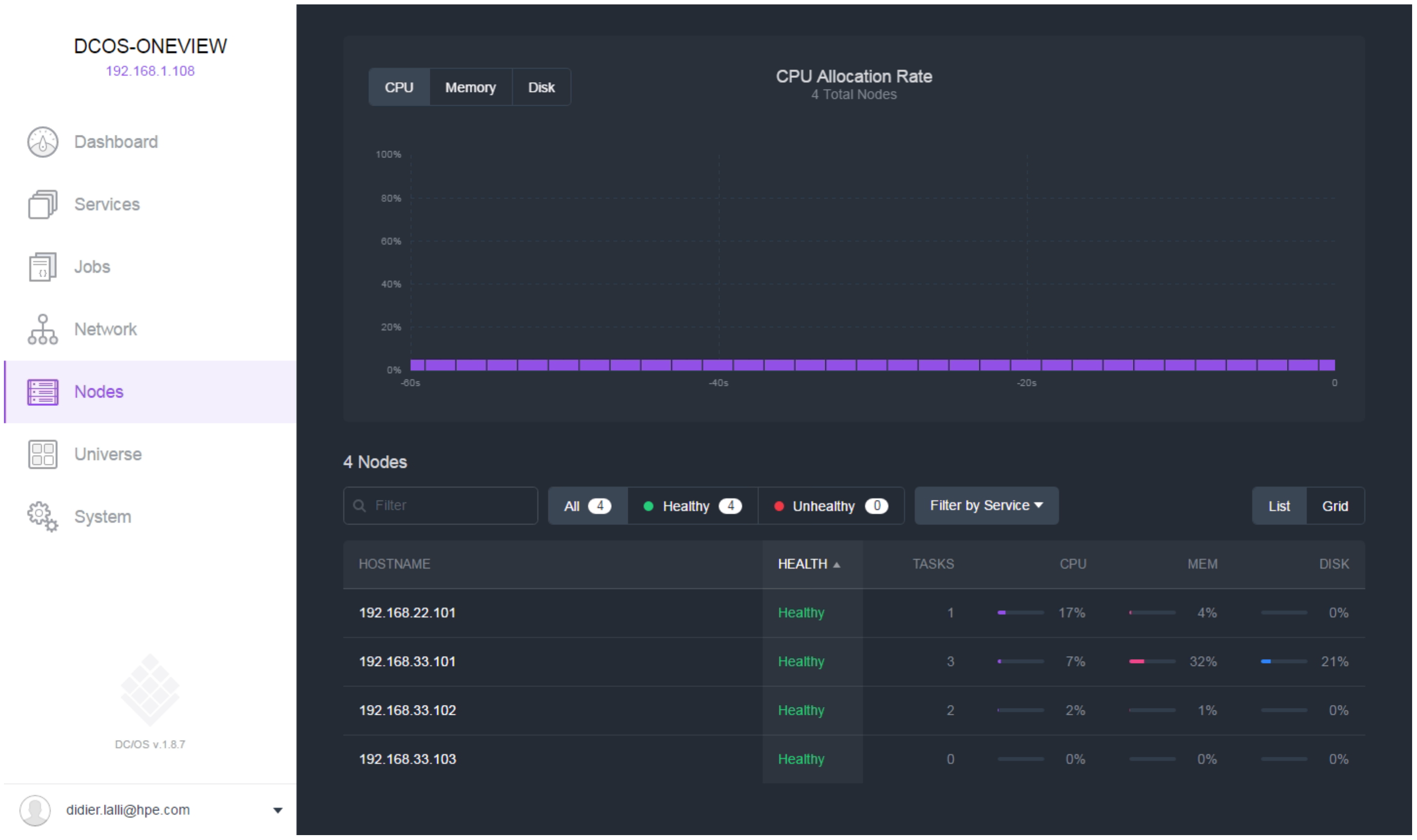
HOSTNAME	HEALTH ▲	TASKS	CPU	MEM	DISK
192.168.22.101	Healthy	1	17%	4%	0%
192.168.33.101	Healthy	3	7%	32%	21%
192.168.33.102	Healthy	2	2%	1%	0%

Use case 2 - Adding an Agent node to a live DC/OS cluster

Because the Ansible playbook have been designed to be idempotent (meaning that when run multiple times it will only make the changes to bring the existing configuration on par with the desired state, as described by the inventory **hosts** file), we can add node in the DC/OS cluster by simply adding new nodes in the hosts file. In our environment, we added an **agent4** to the list of private agents:

```
[dcos-private-agents]
agent2
agent3
agent4
```

And reran the same Ansible playbook, and waited for the end of it to refresh the Nodes view. The 4th agent shows up in the node list as a Healthy node.



HOSTNAME	HEALTH	TASKS	CPU	MEM	DISK
192.168.22.101	Healthy	1	17%	4%	0%
192.168.33.101	Healthy	3	7%	32%	21%
192.168.33.102	Healthy	2	2%	1%	0%
192.168.33.103	Healthy	0	0%	0%	0%

Use case 3 - Removing an Agent node

We have created a separate playbook to remove an agent from the cluster. This playbooks will operate on nodes listed in the dcos-evicted-agent section at the end of the inventory hosts file:

```
[dcos-evicted-agents]
agent4
```

In our case, we are evicting agent 4. So, we should also remove it from the private agents list of that same hosts file:

```
[dcos-private-agents]
agent2
agent3
```

Once this is done we can run the following ansible playbook:

```
ansible-playbook ov_dcos_clean_agent.yml -vvv -e
  "ansible_python_interpreter=/usr/local/bin/python2.7"
```

This will first drain all services of all dcos-evicted-agents and then delete the HPE OneView server profiles associated with these nodes and return those resources to the available compute pool of resources.

Use case 4 - Decomposing and returning resources to free pools

We have created another ansible playbook for destroying completely the DC/OS cluster and returning all the compute resources to the free pool of resources. The playbook is called ov_dcos_decompose_all.yml, and it should be invoked (with care) using the following command:

```
ansible-playbook ov_dcos_decompose_all.yml -vvv -e "ansible_python_interpreter=/usr/local/bin/python2.7"
```

Notes

Be extremely careful with this playbook as no question is asked and the entire cluster will be deleted

Full infrastructure as code

While for this PoC, we have created the server profile templates in HPE OneView using the GUI, we can also use Ansible and yaml to describe those templates and store them in your prefered code management solution. This is particularly attractive in DevOps environments where it is important that everything be managed as code (including the components of the infrastructure). We have provided an example of Ansible playbook to create a Server Profile Template in HPE OneView called create_server_profile_template.yml. These should be adapted to match the target network configuration.

Proof-of-concept lab environment - Synergy (Phase 2)

Compute resource configuration

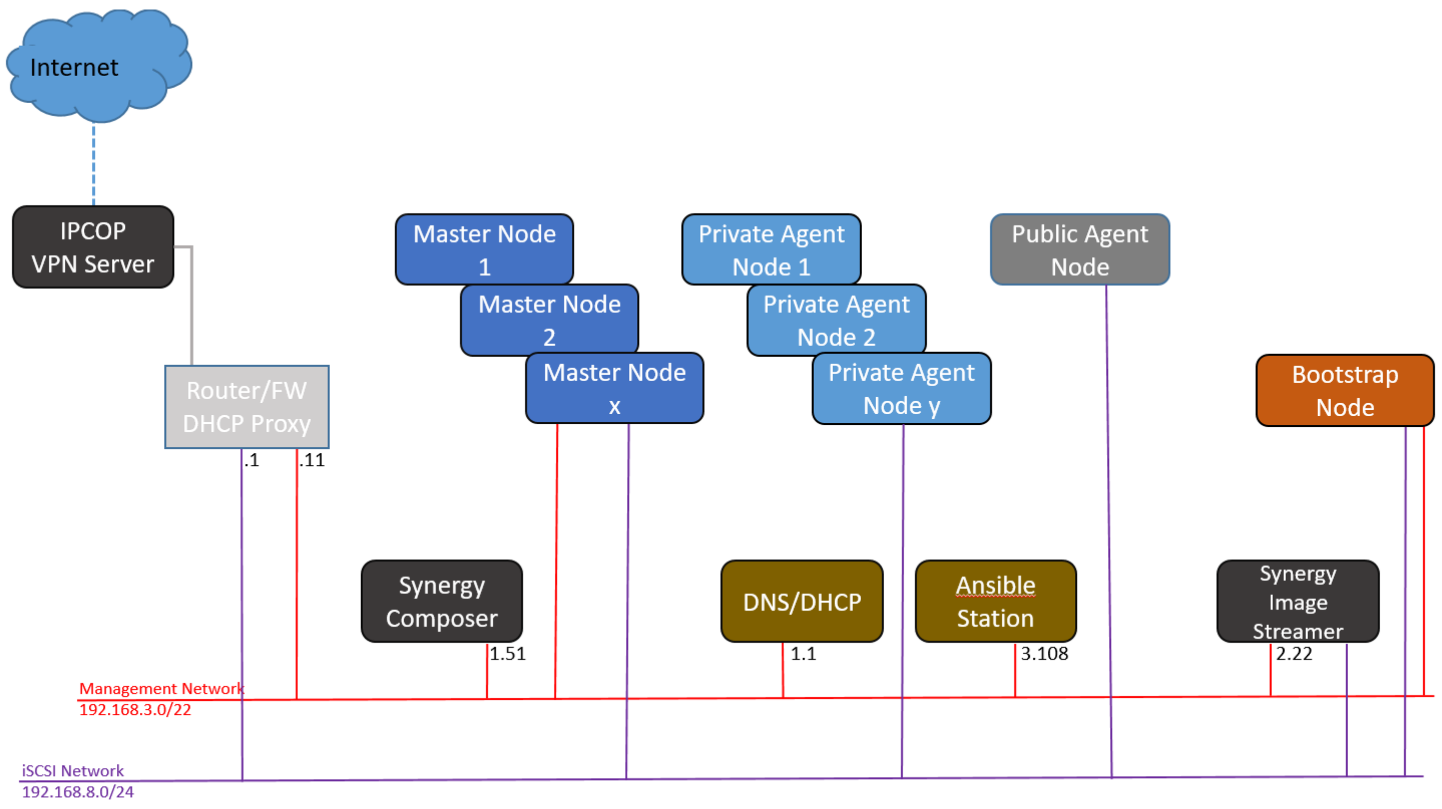
HPE assembled a proof-of-concept lab environment for DCOS which consists of 18xSY480 Gen 9 servers in 3 Synergy frames.

Storage configuration

THE HPE servers were configured to use the Synergy Image Streamer for the OS Volume. The test does not include setting up external storage for DC/OS agents.

Network configuration

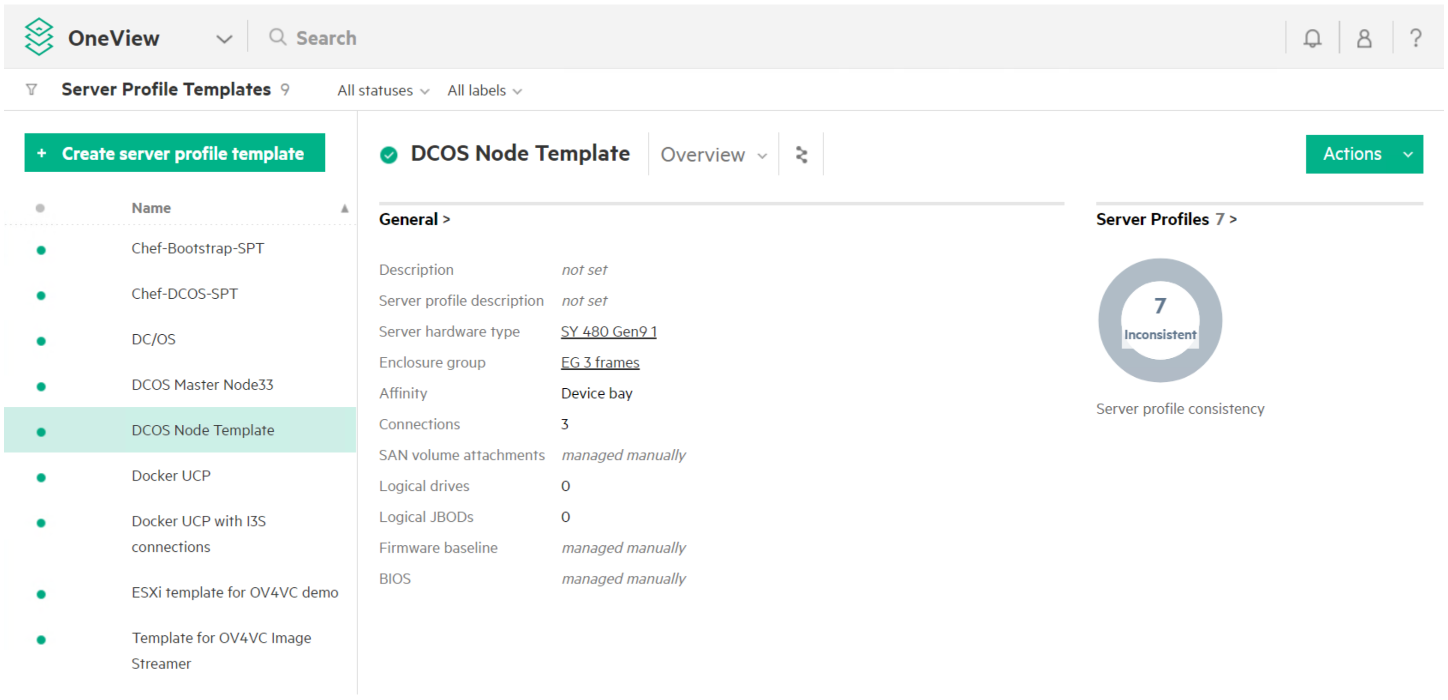
In this second phase a flat network was used, so all nodes are on the same physical network. So the configuration is slightly simplified. Also an iSCSI network is introduced to support Synergy Image Streamer



HPE Composer configuration

In HPE Composer (OneView) we have created a single Server Profile Template for all nodes with the following details:

- 1 network connection
- Server Hardware Type: SY480 Gen9



- Chef-Bootstrap-SPT
- Chef-DCOS-SPT
- DC/OS
- DCOS Master Node33
- DCOS Node Template
- Docker UCP
- Docker UCP with I3S connections
- ESXi template for OV4VC demo
- Template for OV4VC Image Streamer

DCOS Node Template

General >

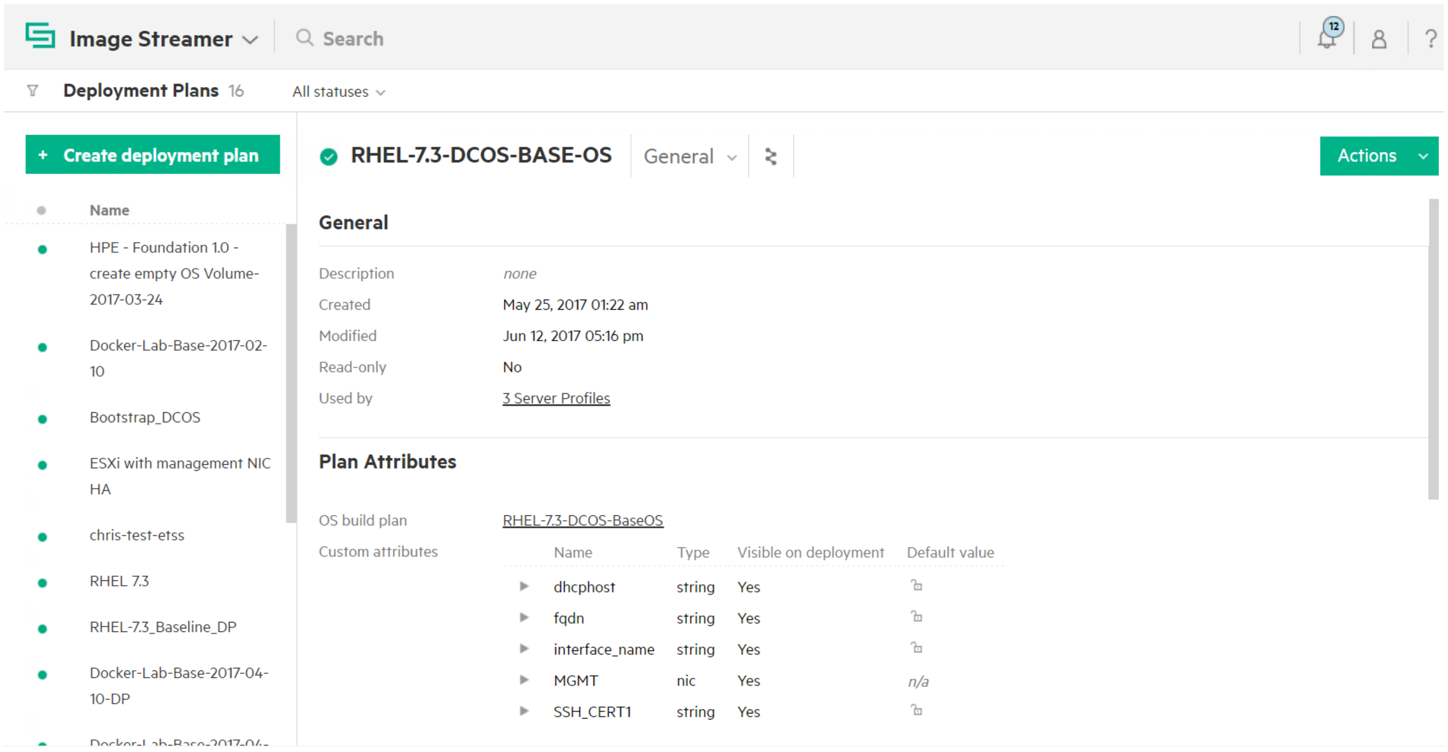
Description	*not set*
Server profile description	*not set*
Server hardware type	SY 480 Gen9 1
Enclosure group	EG 3 frames
Affinity	Device bay
Connections	3
SAN volume attachments	*managed manually*
Logical drives	0
Logical JBODs	0
Firmware baseline	*managed manually*
BIOS	*managed manually*

Server Profiles 7 >

Server profile consistency

HPE Image Streamer configuration

We configured Image Streamer to have an OS Deployment Plan



- HPE - Foundation 1.0 - create empty OS Volume-2017-03-24
- Docker-Lab-Base-2017-02-10
- Bootstrap_DCOS
- ESXi with management NIC HA
- chris-test-etss
- RHEL 7.3
- RHEL-7.3_Baseline_DP
- Docker-Lab-Base-2017-04-10-DP
- Docker-Lab-Base-2017-04-

RHEL-7.3-DCOS-BASE-OS

General

Description	*none*
Created	May 25, 2017 01:22 am
Modified	Jun 12, 2017 05:16 pm
Read-only	No
Used by	3 Server Profiles

Plan Attributes

OS build plan: RHEL-7.3-DCOS-BaseOS

Custom attributes

Name	Type	Visible on deployment	Default value
dhcphost	string	Yes	
fqdn	string	Yes	
interface_name	string	Yes	
MGMT	nic	Yes	*n/a*
SSH_CERT1	string	Yes	

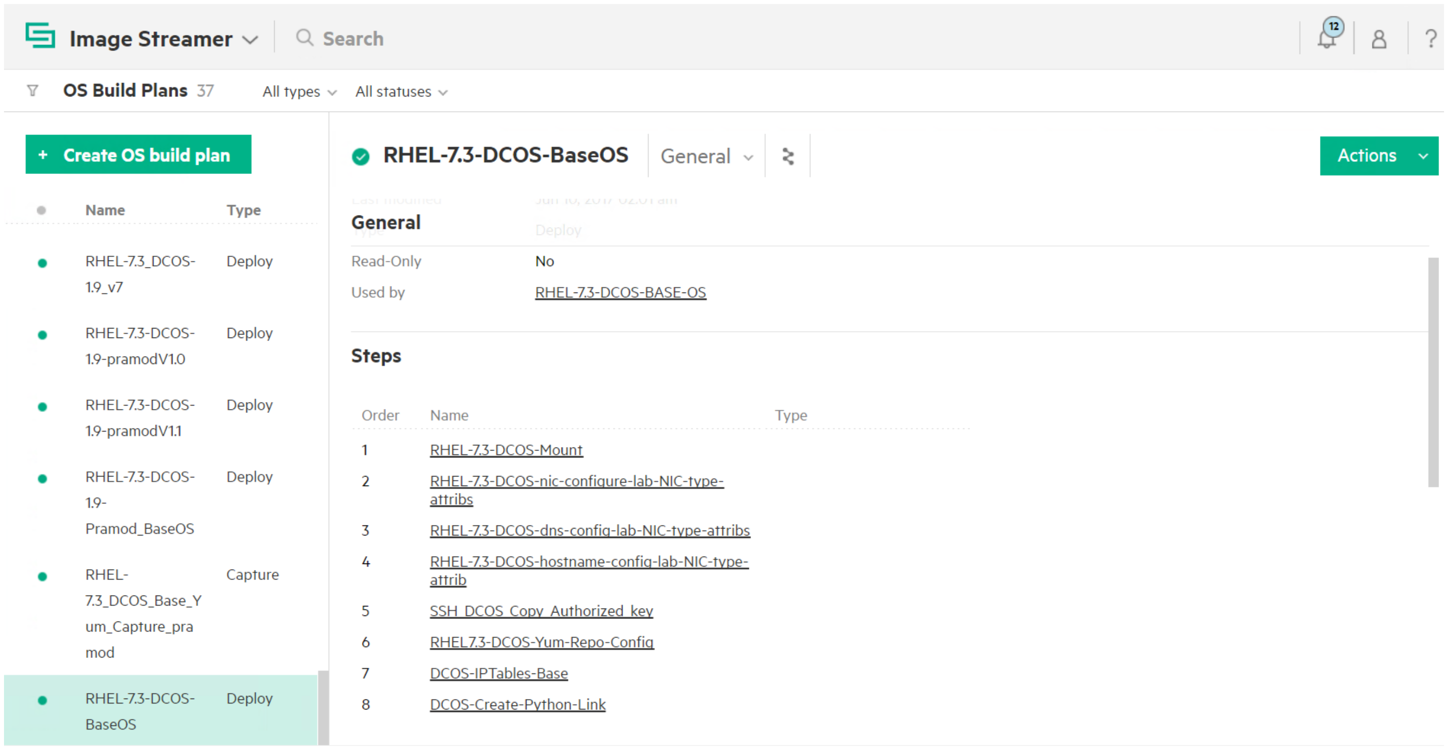
This OS Deployment plan has a number of attributes:

dhcphost and fqdn to pass the hostname and the fully qualified name of the system to provision. MGMT and interface_name to pass the interface used for networking, SSH_CERT1 and SSH_CERT2 for the SSH key used by Ansible.

Notes:

This is a temporary workaround as there is a current limitation on the length of a plan attribute.

The OS Build plan is referencing a golden image and an OS build plan. The OS Build Plan has the steps which will happen to transform a Golden Image into a dedicated OS Volume. In most case this steps are applying changes on the OS Volume, using the input parameters of the Build Plan (hostname, SSH Key).



Name	Type
RHEL-7.3_DCOS-1.9_v7	Deploy
RHEL-7.3-DCOS-1.9-pramodV1.0	Deploy
RHEL-7.3-DCOS-1.9-pramodV1.1	Deploy
RHEL-7.3-DCOS-1.9-Pramod_BaseOS	Deploy
RHEL-7.3_DCOS_Base_Yum_Capture_pramod	Capture
RHEL-7.3-DCOS-BaseOS	Deploy

General

Read-Only	No
Used by	RHEL-7.3-DCOS-BASE-OS

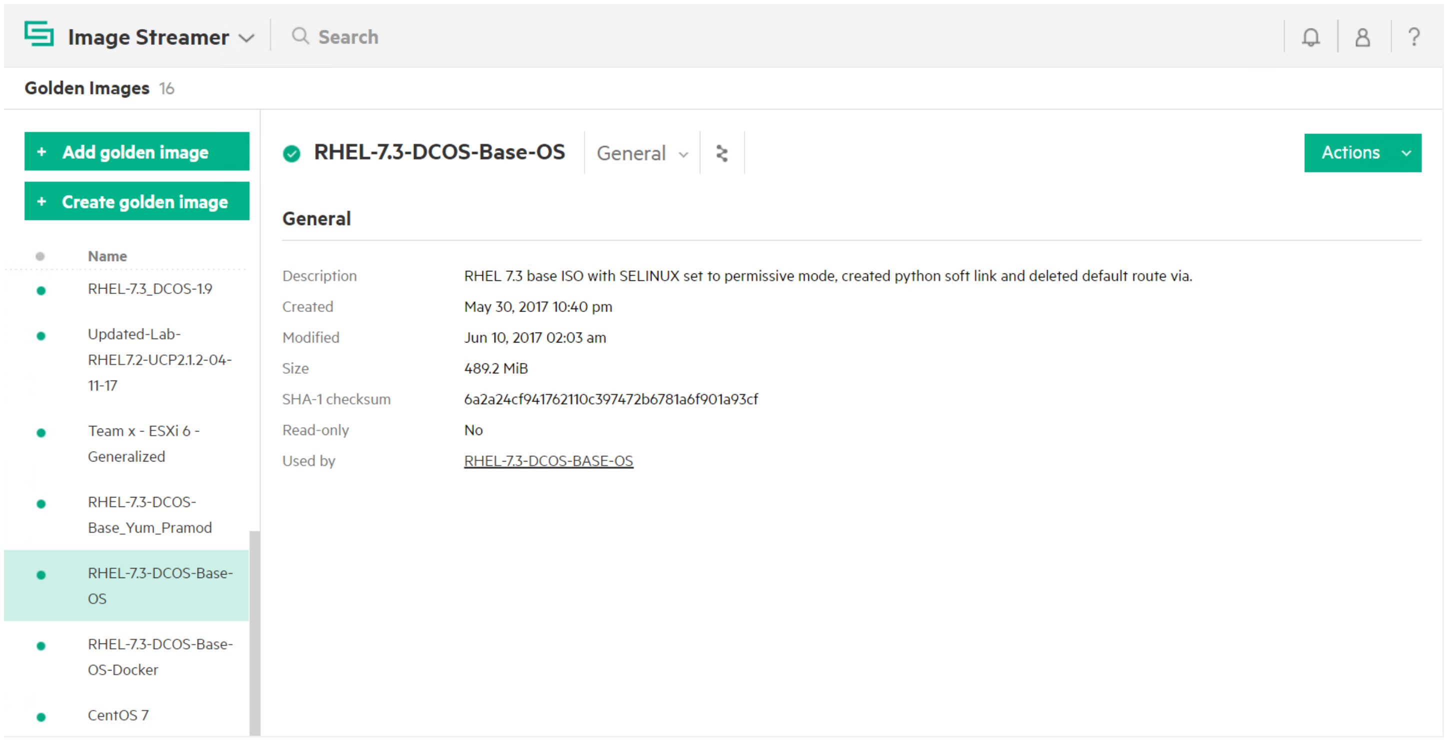
Steps

Order	Name	Type
1	RHEL-7.3-DCOS-Mount	
2	RHEL-7.3-DCOS-nic-configure-lab-NIC-type-attribs	
3	RHEL-7.3-DCOS-dns-config-lab-NIC-type-attribs	
4	RHEL-7.3-DCOS-hostname-config-lab-NIC-type-attrib	
5	SSH_DCOS_Copy_Authorized_key	
6	RHEL7.3-DCOS-Yum-Repo-Config	
7	DCOS-IPTables-Base	
8	DCOS-Create-Python-Link	

Image Streamer Golden Image

We created a Golden image for this test, which contains the following:

- RedHat Enterprise Linux 7.3
- Yum configured to point to a RHEL7.3 repo



General

Description	RHEL 7.3 base ISO with SELINUX set to permissive mode, created python soft link and deleted default route via.
Created	May 30, 2017 10:40 pm
Modified	Jun 10, 2017 02:03 am
Size	489.2 MiB
SHA-1 checksum	6a2a24cf941762110c397472b6781a6f901a93cf
Read-only	No
Used by	RHEL-7.3-DCOS-BASE-OS

Notes:
A similar golden image should be built and referenced by the OS Deployment Plan provided in the artifact bundle of this solution before it can be reused.

Importing the solution artifact bundle

We created an artifact bundle with the following components:

- 1 x OS build plan
- 8 x Plan Scripts used by the OS build plan

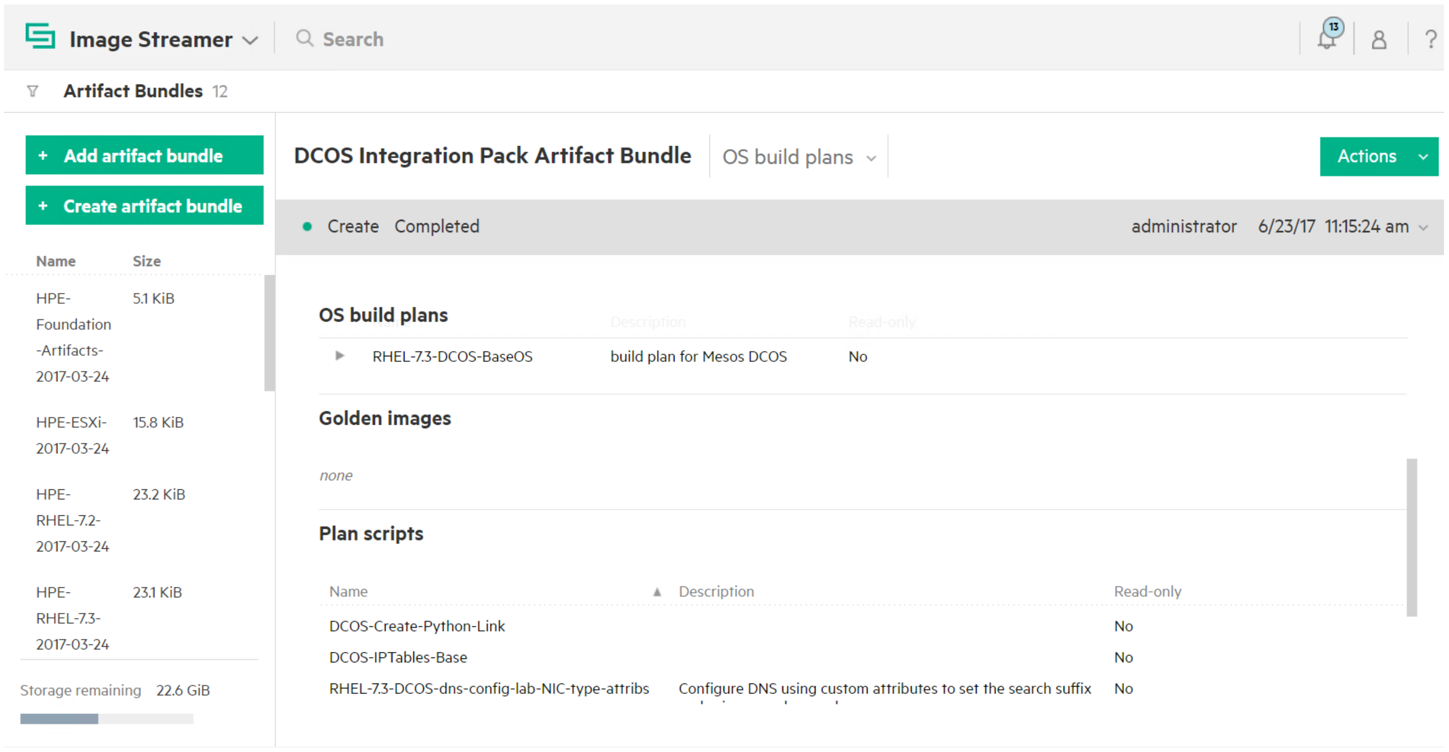


Image Streamer
Search
Artifact Bundles 12
+ Add artifact bundle
+ Create artifact bundle

Name	Size
HPE-Foundation-Artifacts-2017-03-24	5.1 KiB
HPE-ESXi-2017-03-24	15.8 KiB
HPE-RHEL-7.2-2017-03-24	23.2 KiB
HPE-RHEL-7.3-2017-03-24	23.1 KiB

Storage remaining 22.6 GiB

DCOS Integration Pack Artifact Bundle
OS build plans
Actions
● Create Completed administrator 6/23/17 11:15:24 am

OS build plans

	Description	Read-only
RHEL-7.3-DCOS-BaseOS	build plan for Mesos DCOS	No

Golden images

none

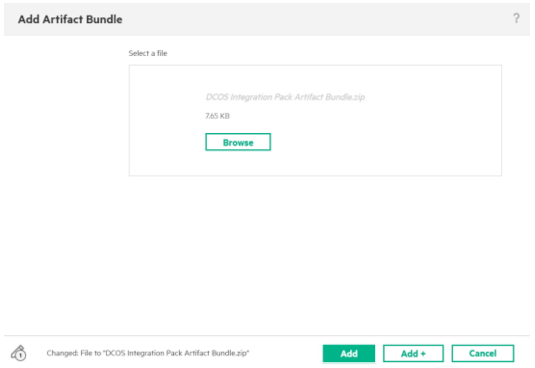
Plan scripts

Name	Description	Read-only
DCOS-Create-Python-Link		No
DCOS-IPTables-Base		No
RHEL-7.3-DCOS-dns-config-lab-NIC-type-attribs	Configure DNS using custom attributes to set the search suffix	No

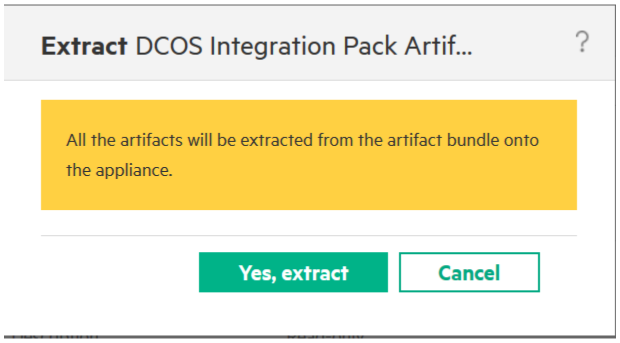
In order to use this solution, you should:

1. Download the artifact bundle from [Github]()

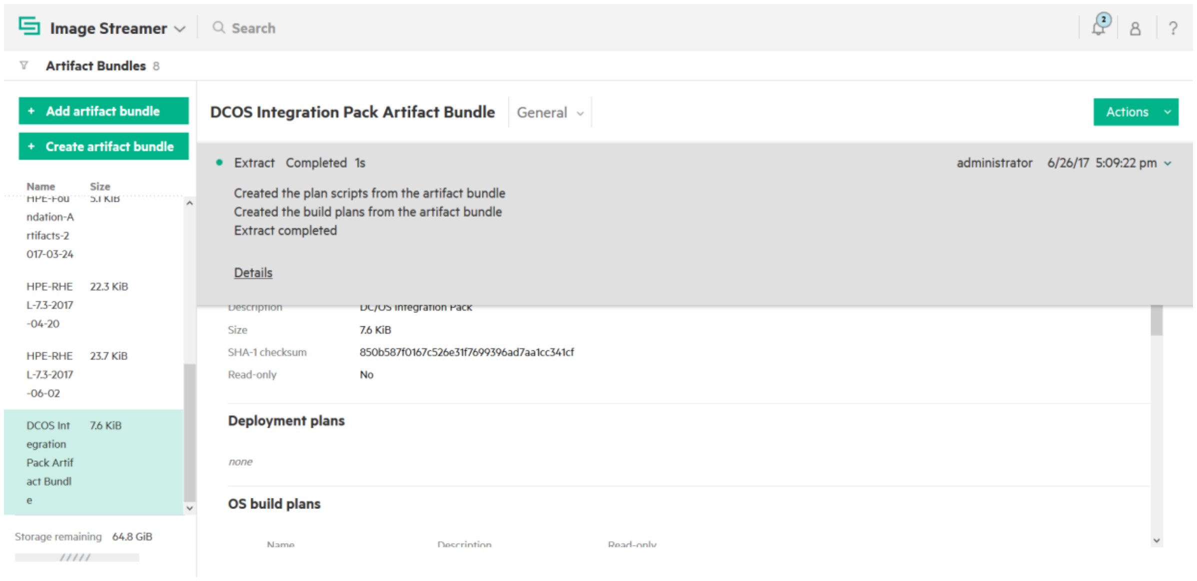
2. Import the artifact bundle into Image Streamer


Add Artifact Bundle
Select a file
DCOS Integration Pack Artifact Bundle.zip
7.65 KB
Browse
Changed: File to "DCOS Integration Pack Artifact Bundle.zip"
Add
Add +
Cancel

3. Extract components from bundle (1 OS Build plan and 8 plan scripts)



All the artifacts will be extracted from the artifact bundle onto the appliance.



DCOS Integration Pack Artifact Bundle

Extract Completed 1s administrator 6/26/17 5:09:22 pm

Created the plan scripts from the artifact bundle
Created the build plans from the artifact bundle
Extract completed

Name	Size
HPE-Foundation-Artifacts-2017-03-24	5.1 KiB
HPE-RHEL-7.3-2017-04-20	22.3 KiB
HPE-RHEL-7.3-2017-06-02	23.7 KiB
DCOS Integration Pack Artifact Bundle	7.6 KiB

Storage remaining 64.8 GiB

Description	DC/OS Integration Pack
Size	7.6 KiB
SHA-1 checksum	850b587f0167c526e31f7699396ad7aa1cc341cf
Read-only	No

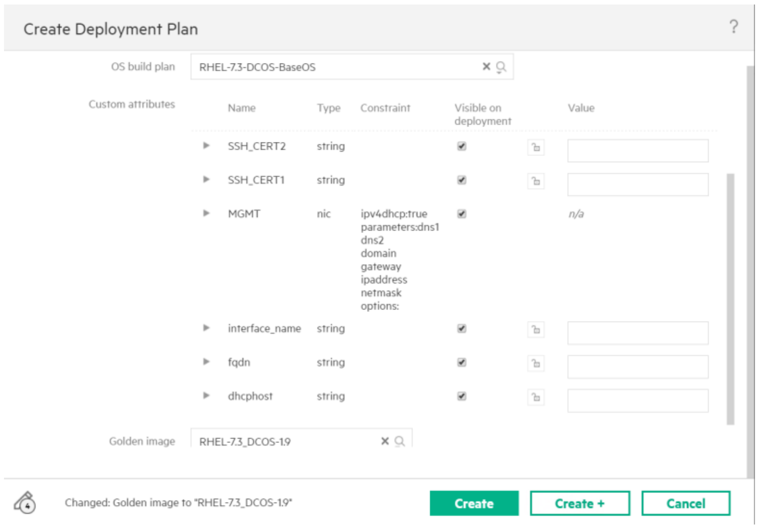
Deployment plans

none

OS build plans

Name Description Read-only

4. Build appropriate golden image following instructions provided by this whitepaper
5. Create a Deployment Plan which associates the imported OS build plan and the created golden image. Leave all attribute values blank



Create Deployment Plan

OS build plan: RHEL-7.3-DCOS-BaseOS

Custom attributes

Name	Type	Constraint	Visible on deployment	Value
SSH_CERT2	string		✓	
SSH_CERT1	string		✓	
MGMT	nic	ipv4dhcp:true parameters:dns1 dns2 domain gateway ipaddress netmask options:	✓	*n/a*
interface_name	string		✓	
fqdn	string		✓	
dhcphost	string		✓	

Golden image: RHEL-7.3_DCOS-1.9

Changed: Golden image to "RHEL-7.3_DCOS-1.9"

Use case 1 - Day 0 DC/OS Deployment

We used the same Ansible playbook, but adapted some of the files only when the architecture imposed a different approach. To do this we used a new variable in group_vars/all called HARDWARE_ARCHITECTURE which should be set to either

- HPE-CCLASS
- HPE-SYNERGY
- HPE-DL

Then we provided a custom file for each architecture, for example: docker-on-HPE-SYNERGY.yml and docker-on-HPE-CCLASS.yml. Finally within Ansible, we have a single playbook which references: docker-on-{{ HARDWARE_ARCHITECTURE }}.yml

Notes:

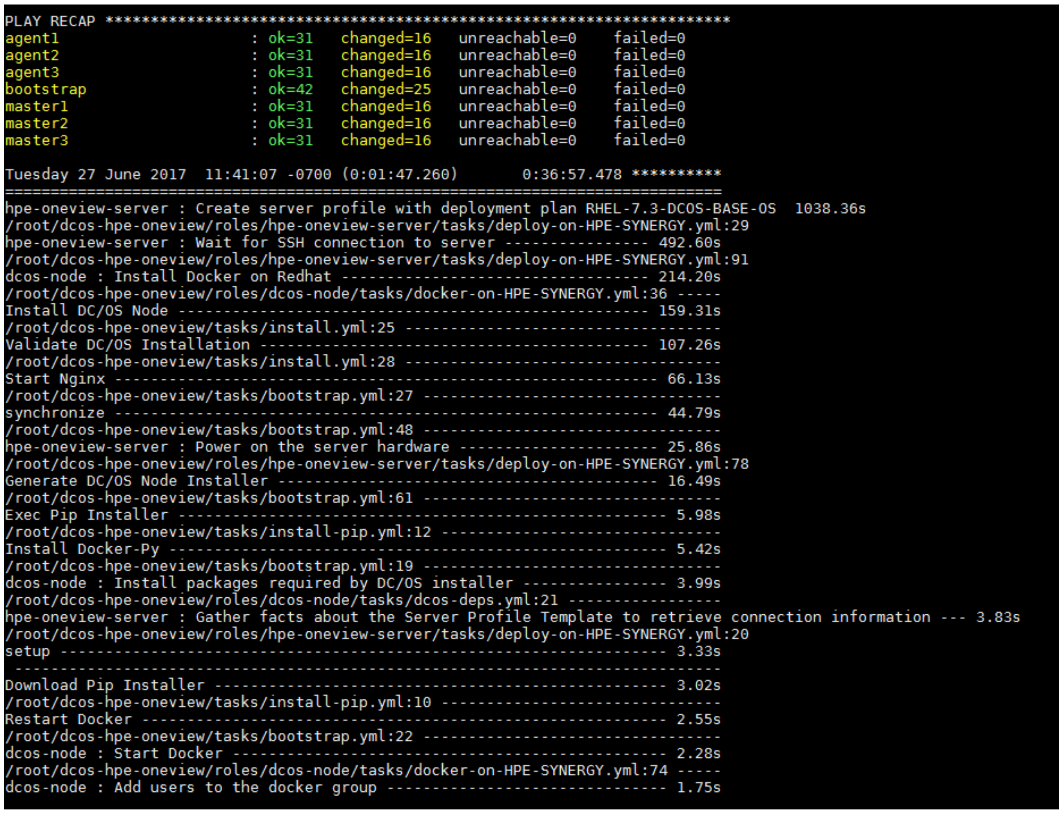
We provided implementation for HPE-CCLASS and HPE-SYNERGY but not yet HPE-DL.

We also added a few more variables in group_vars/all:

- INTERFACE_NAME: ens3f4 : hold the name of the interface to be used for public networking (and SSH traffic used by Ansible)
- DCOS_BOOTSTRAP_URL: 'http://bootstrap.{{ domain_name }}' : holds the fqdn of the bootstrap machine
- management_network_name: 'Mgmt A' : holds the name of the Management Network in the OneView Server profile

We can invoke the playbook the same way as in Phase 1:

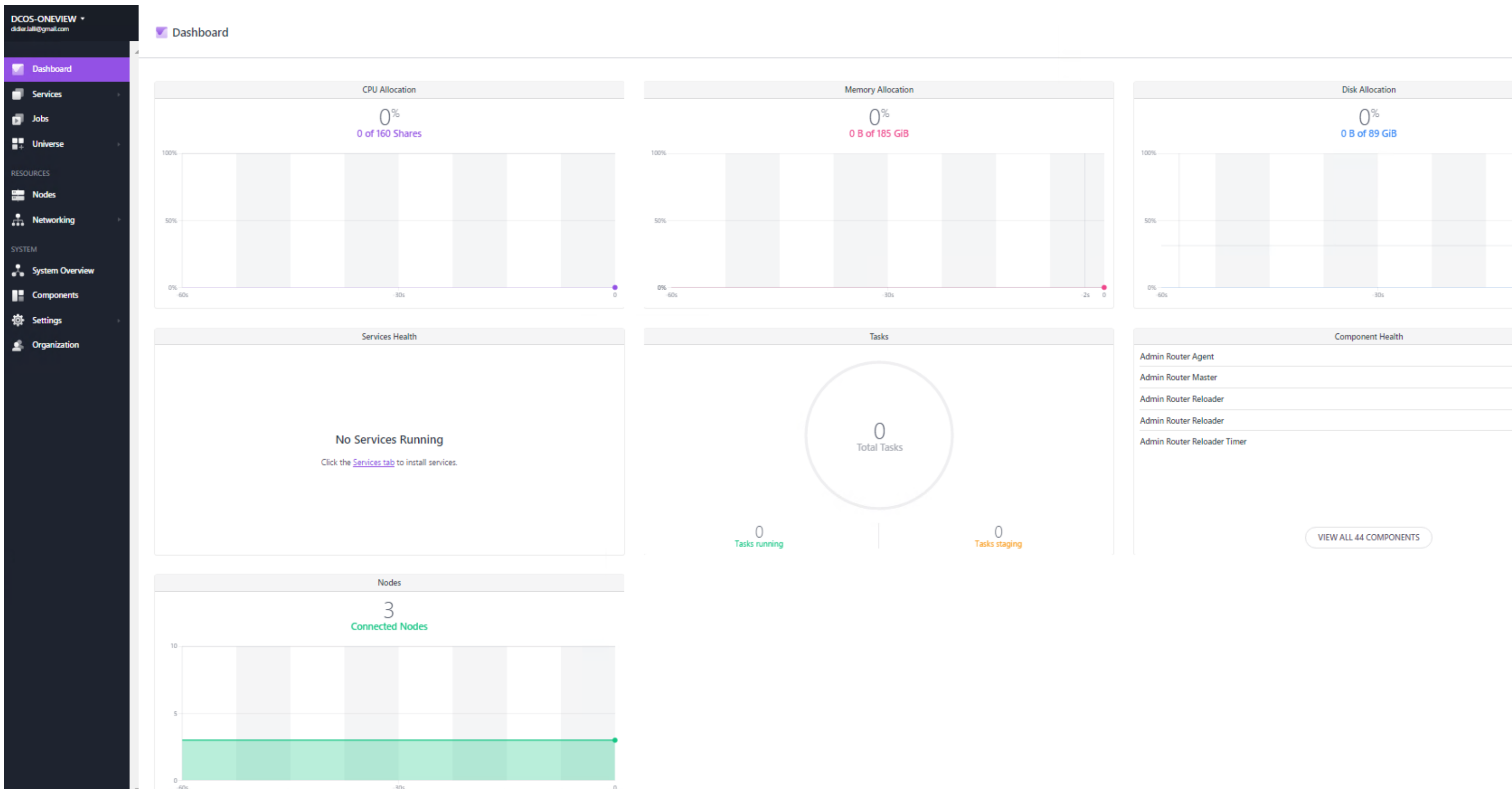
```
ansible-playbook ov_dcos.yml -vvv -e "ansible_python_interpreter=/usr/local/bin/python2.7"
```



```
PLAY RECAP *********************************************************************
agent1                     : ok=31    changed=16    unreachable=0    failed=0
agent2                     : ok=31    changed=16    unreachable=0    failed=0
agent3                     : ok=31    changed=16    unreachable=0    failed=0
bootstrap                  : ok=42    changed=25    unreachable=0    failed=0
master1                    : ok=31    changed=16    unreachable=0    failed=0
master2                    : ok=31    changed=16    unreachable=0    failed=0
master3                    : ok=31    changed=16    unreachable=0    failed=0

Tuesday 27 June 2017  11:41:07 -0700 (0:01:47.260)       0:36:57.478 **********
===============================================================================
hpe-oneview-server : Create server profile with deployment plan RHEL-7.3-DCOS-BASE-OS  1038.36s
/root/dcos-hpe-oneview/roles/hpe-oneview-server/tasks/deploy-on-HPE-SYNERGY.yml:29
hpe-oneview-server : Wait for SSH connection to server ---------------- 492.60s
/root/dcos-hpe-oneview/roles/hpe-oneview-server/tasks/deploy-on-HPE-SYNERGY.yml:91
dcos-node : Install Docker on Redhat ---------------------------------- 214.20s
/root/dcos-hpe-oneview/roles/dcos-node/tasks/docker-on-HPE-SYNERGY.yml:36 -----
Install DC/OS Node ---------------------------------------------------- 159.31s
/root/dcos-hpe-oneview/tasks/install.yml:25 -----------------------------------
Validate DC/OS Installation ------------------------------------------- 107.26s
/root/dcos-hpe-oneview/tasks/install.yml:28 -----------------------------------
Start Nginx ------------------------------------------------------------ 66.13s
/root/dcos-hpe-oneview/tasks/bootstrap.yml:27 ---------------------------------
synchronize ------------------------------------------------------------ 44.79s
/root/dcos-hpe-oneview/tasks/bootstrap.yml:48 ---------------------------------
hpe-oneview-server : Power on the server hardware ---------------------- 25.86s
/root/dcos-hpe-oneview/roles/hpe-oneview-server/tasks/deploy-on-HPE-SYNERGY.yml:78
Generate DC/OS Node Installer ------------------------------------------ 16.49s
/root/dcos-hpe-oneview/tasks/bootstrap.yml:61 ---------------------------------
Exec Pip Installer ------------------------------------------------------ 5.98s
/root/dcos-hpe-oneview/tasks/install-pip.yml:12 -------------------------------
Install Docker-Py ------------------------------------------------------- 5.42s
/root/dcos-hpe-oneview/tasks/bootstrap.yml:19 ---------------------------------
dcos-node : Install packages required by DC/OS installer ---------------- 3.99s
/root/dcos-hpe-oneview/roles/dcos-node/tasks/dcos-deps.yml:21 -----------------
hpe-oneview-server : Gather facts about the Server Profile Template to retrieve connection information --- 3.83s
/root/dcos-hpe-oneview/roles/hpe-oneview-server/tasks/deploy-on-HPE-SYNERGY.yml:20
setup ------------------------------------------------------------------- 3.33s
 ------------------------------------------------------------------------------
Download Pip Installer -------------------------------------------------- 3.02s
/root/dcos-hpe-oneview/tasks/install-pip.yml:10 -------------------------------
Restart Docker ---------------------------------------------------------- 2.55s
/root/dcos-hpe-oneview/tasks/bootstrap.yml:22 ---------------------------------
dcos-node : Start Docker ------------------------------------------------ 2.28s
/root/dcos-hpe-oneview/roles/dcos-node/tasks/docker-on-HPE-SYNERGY.yml:74 -----
dcos-node : Add users to the docker group ------------------------------- 1.75s
```

Provisioning was down from 1h22mn to 36mn using the ImageStream instead of a PXE boot installation such as ICsp.

We can now connect to the dashboard of our newly provisioned Mesosphere DC/OS Cluster:



0% 0 of 160 Shares

0% 0 B of 185 GiB

0% 0 B of 89 GiB

No Services Running

Click the Services tab to install services.

0 Total Tasks

0 Tasks running

0 Tasks staging

Admin Router Agent

Admin Router Master

Admin Router Reloader

Admin Router Reloader

Admin Router Reloader Timer

3 Connected Nodes

Notes:
Use cases 2 (adding nodes) and 3 (removing agent nodes) were also tested with Synergy, and worked exactly the same way as described in Phase 1.

Summary

This solution leverages HPE OneView and HPE Composable Infrastructure to provision a Mesosphere DC/OS Cluster, ready to run payload. The solution can provision the cluster from a set of available compute resources in an HPE Composable Infrastructure, in less than 90mn using Ansible (C-Class) and even less than TBD minutes (Synergy and ImageStreamer). The solution is fully automated and consistent, allowing to truly manage your infrastructure as code. The solution can be used as an example to build other solutions.