HPE Express Containers with Docker Enterprise Edition: Dev Edition

Deployment Guide and Best Practices

Revision 1.0 – December 13, 2017

Deployment guide

Contents

Introduction

HPE Express Containers with Docker Enterprise Edition (EE) is a complete solution from Hewlett Packard Enterprise that includes all the hardware, software, professional services, and support you need to deploy a containers-as-a-service (CaaS) platform, allowing you to get up and running quickly and efficiently. The solution takes the HPE hyperconverged infrastructure and combines it with Docker's enterprise-grade container platform, popular open source tools, along with deployment and advisory services from HPE Pointnext.

HPE Express Containers with Docker EE is ideal for customers migrating legacy applications to containers, transitioning to a container DevOps development model or needing a hybrid environment to support container and non-containerized applications on a common VM platform. HPE Express Containers with Docker EE provides a solution for both IT development and IT operations, and comes in two versions. The version for IT developers (HPE Express Containers with Docker EE: Dev Edition) addresses the need to provide a cloud-like container development environment with built-in container tooling. The version for IT operations (HPE Express Containers with Docker EE: Ops Edition) addresses the need to have a production ready environment that is very easy to deploy and manage.

This document describes the best practices for deploying and operating the HPE Express Containers with Docker EE: Dev Edition. It describes how to automate the provisioning of the environment using a set of Ansible playbooks. It also outlines a set of manual steps to harden, secure and audit the overall status of the system, and deploys the ELK, Play-with-Docker, and CloudBees Jenkins tools. A corresponding document focused on setting up HPE Express Containers with Docker: Ops Edition is also available.

Note

The Ansible playbooks described in this document are only intended for Day 0 deployment automation of Docker EE on HPE SimpliVity.

The Ansible playbooks described in this document are not directly supported by HPE and are intended as an example of deploying Docker EE on HPE SimpliVity. We welcome input from the user community via GitHub to help us prioritize all future bug fixes and feature enhancements.

About Ansible

Ansible is an open-source automation engine that automates software provisioning, configuration management, and application deployment.

As with most configuration management software, Ansible has two types of servers: the controlling machine and the nodes. A single controlling machine orchestrates the nodes by deploying modules to the nodes over SSH. The modules are temporarily stored on the nodes and communicate with the controlling machine through a JSON protocol over the standard output. When Ansible is not managing nodes, it does not consume resources because no daemons or programs are executing for Ansible in the background. Ansible uses one or more inventory files to manage the configuration of the multiple nodes in the system.

In contrast with other popular configuration management software, such as Chef, Puppet, and CFEngine, Ansible uses an agentless architecture. With an agent-based architecture, nodes must have a locally installed daemon that communicates with a controlling machine. With an agentless architecture, nodes are not required to install and run background daemons to connect with a controlling machine. This type of architecture reduces the overhead on the network by preventing the nodes from polling the controlling machine.

More information about Ansible can be found at: http://docs.ansible.com

About Docker Enterprise Edition

Docker Enterprise Edition (EE) is the leading enterprise containers-as-a-service (CaaS) software platform for IT that manages and secures diverse applications across disparate infrastructure, both on-premises and in the cloud. Docker EE provides integrated container management and security from development to production. Enterprise-ready capabilities like multi-architecture orchestration and secure software supply chain give IT teams the ability to manage and secure containers without breaking the developer experience.

Docker EE provides:

- Integrated management of all application resources from a single web admin UI.
- Frictionless deployment of applications and Compose files to production in a few clicks.

- Multi-tenant system with granular role-based access control (RBAC) and LDAP/AD integration.
- Self-healing application deployment with the ability to apply rolling application updates.
- End-to-end security model with secrets management, image signing and image security scanning.

More information about Docker Enterprise Edition can be found at: https://www.docker.com/enterprise-edition

About HPE SimpliVity

HPE SimpliVity is an enterprise-grade hyper-converged platform combining best-in-class data services with the world's best-selling server.

Rapid proliferation of applications and the increasing cost of maintaining legacy infrastructure causes significant IT challenges for many organizations. With HPE SimpliVity, you can streamline and enable IT development at a fraction of the cost of traditional and public cloud solutions by combining your IT infrastructure and advanced data services into a single, integrated solution. HPE SimpliVity is a powerful, simple, and efficient hyper-converged platform that joins best-in-class data services with the world's best-selling server and offers the industry's most complete guarantee.

More information about HPE SimpliVity can be found at: https://www.hpe.com/us/en/integrated-systems/simplivity.html

Target Audience: This document is primarily aimed at technical individuals working in the development side of the pipeline, such as technical architects, developers and system administrators, but anybody with an interest in automating the provisioning of virtual servers and containers may find this document useful.

Assumptions: This document assumes a minimum understanding of concepts like virtualization, containerization and some knowledge around Linux® and VMWare® technologies.

Required Versions: The following software versions were used to test the playbooks that are described in later sections. Other versions may work but have not been tested.

- Ansible 2.2 and 2.3. Please note that the playbooks will not work with Ansible 2.4 due to an open defect https://github.com/ansible/ansible/issues/32000. Do not use Ansible 2.4 until this defect is fixed.
- Docker EE 17.06 (tested with UCP 2.2.3 and 2.2.4 and DTR 2.4.0)
- Red Hat® Enterprise Linux 7.3 and 7.4
- VMWare ESXi 6.5.0 and vCenter 6.5.0
- HPE SimpliVity OmniStack 3.7.1.60

Architecture

The development environment is comprised of two HPE SimpliVity 380 Gen10 servers. HPE recommends dual socket SimpliVity systems with at least 14 CPU cores per socket (28 total cores per system) for optimal performance and support during HA failover scenario. Please refer to Appendix A for a sample BOM. Since the SimpliVity technology relies on VMware virtualization, the servers are managed using vCenter. The load among the two hosts will be shared as per Figure 1.

Uptime is paramount for any users implementing Docker containers in business critical environments. HPE Express Containers with Docker EE offers various levels of high availability (HA) to support continuous availability. All containers including the Docker system containers are protected by Docker's swarm mode. Swarm mode can protect against individual hardware, network, and container failures based on the user's declarative model.

HPE Express Containers with Docker EE also deploys load balancers in the system to help with container traffic management. There are three load balancer VMs – UCP load balancer, DTR load balancer, and Docker worker node load balancer. The UCP load balancer and the DTR load balancer have a minimal effect on the current configuration because only a single instance of both UCP and DTR are deployed. These load balancers are included in case the user decides to setup additional UCP and DTR nodes in the future to improve high availability.

Since the load balancers exist in VMs, they have some degree of HA but may incur some downtime during the restoration of these VMs due to a planned or unplanned outage. For optimal HA configuration, the user should consider implementing a HA load balancer architecture using the Virtual Router Redundancy Protocol (VRRP). For more information see http://www.haproxy.com/solutions/high-availability/.

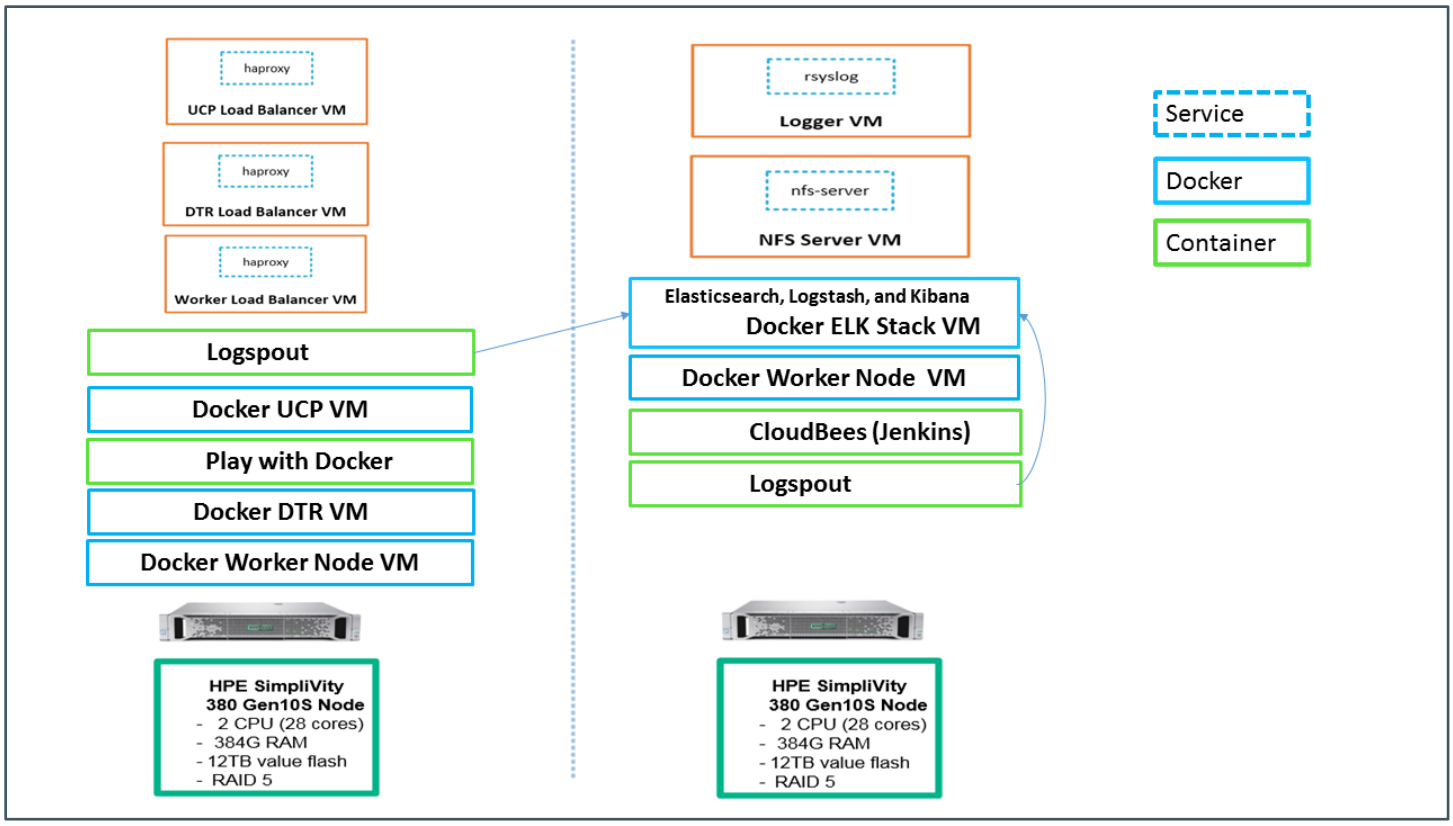

Figure 1. Solution architecture

The Ansible playbooks can be modified to fit your environment and your high availability (HA) needs. By default, the Ansible Playbooks will set up a 2 node environment. HPE and Docker recommend a minimal starter configuration of 2 physical nodes for running Docker in a development environment.

The distribution of the Docker and non-Docker modules over the 2 physical nodes via virtual machines (VMs) is as follows:

- 1 Docker Universal Control Plane (UCP) VM node
- 1 Docker Trusted Registry (DTR) VM node
- 2 Docker Swarm worker VM nodes for container workloads
- 1 Docker UCP load balancer VM to ensure access to UCP in the event of a node failure
- 1 Docker DTR load balancer VM to ensure access to DTR in the event of a node failure
- 1 Docker Swarm Worker node VM load balancer
- 1 Logging server VM for central logging

- 1 NFS server VM for storage Docker DTR images

In addition to the above, the playbooks also set up:

- Docker persistent storage driver from VMware
- Elasticsearch, Logstash, and Kibana (ELK) stack
- Play with Docker (PWD)
- CloudBees Jenkins
- SimpliVity backup policy for data volumes and for the NFS storage used by DTR for storing Docker images

These instances can live in any of the hosts and they are not redundant. The vSphere Docker volume plug-in stores data in a shared datastore that can be accessed from any machine in the cluster.

Sizing considerations

A node is a machine in the cluster (virtual or physical) with Docker Engine running on it. When provisioning each node, assign it a role: UCP Controller, DTR, or worker node so that it is protected from running application workloads.

To decide what size the node should be in terms of CPU, RAM, and storage resources, consider the following:

1. All nodes should at least fulfil the minimal requirements, for UCP 2.0, 2GB of RAM and 3GB of storage. More detailed requirements are in the UCP documentation.
2. UCP Controller nodes should be provided with more than the minimal requirements, but won't need much more if nothing else runs on them.
3. Ideally, worker node size will vary based on your workloads so it is impossible to define a universal standard size.
4. Other considerations like target density (average number of containers per node), whether one standard node type or several are preferred, and other operational considerations might also influence sizing.

If possible, node size should be determined by experimentation and testing actual workload;, and they should be refined iteratively. A good starting point is to select a standard or default machine type in your environment and use this size only. If your standard machine type provides more resources than the UCP Controllers need, it makes sense to have a smaller node size for these. Whatever the starting choice, it is important to monitor resource usage and cost to improve the model.

For HPE Express Containers with Docker EE: Dev Edition, the following tables describe sizing configurations. The vCPU allocations are described in Table 1 while the memory allocation is described in Table 2.

Table 1. vCPU

vCPUs	simply01	simply02
ucp1	4	
dtr1	2	
worker1	4	
worker2		4
ucb_lb	2	
dtr_lb	2	
worker_lb	2	
nfs		2

logger		2
Total vCPU per node	**16**	**8**

Note

In the case of one ESX host failure, the remaining node can accommodate the amount of vCPU required.

Table 2. Memory allocation (GB)

RAM (GB)	**simply01**	**simply02**
ucp1	8	
dtr1	16	
worker1	16	
worker2		16
ucb_lb	2	
dtr_lb	2	
worker_lb	2	
nfs		2
logger		2
Total RAM required (per node)	**46**	**20**
Total RAM required	**66**	
Available RAM	384	384

Note

In the case of one ESX host failure, the remaining host can accommodate the amount of RAM required for all VMs.

Provisioning the development environment

This section describes in detail how to provision the environment described previously in the architecture section. The high level steps this guide will take are shown in Figure 2.

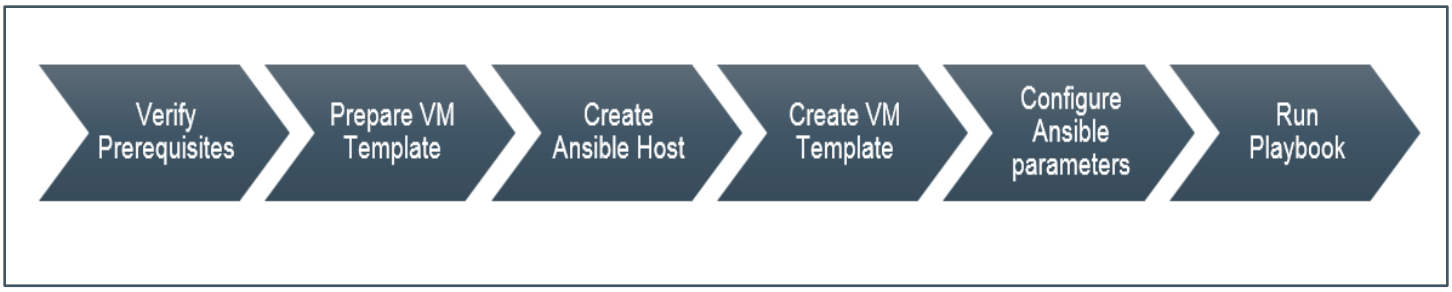

Figure 2. Provisioning steps

Verify Prerequisites

You need assemble the information required to assign values for each and every variable used by the playbooks before you start deployment. The variables are fully documented in the following sections "Editing the group variables" and "Editing the vault". A summary of the information required is presented in Table 3.

Table 3 Summary of information required

Component	Details
Virtual Infrastructure	The FQDN of your vCenter server and the name of the Datacenter which contains the SimpliVity cluster. You will also need administrator credentials in order to create templates, and spin up virtual machines.
SimpliVity Cluster	The name of the SimpliVity cluster and the names of the member of this cluster as they appear in vCenter. You will also need to know the name of the SimpliVity datastore where you want to land the various virtual machines. You may have to create this datastore if you just installed your SimpliVity cluster. In addition, you will need the IP addresses of the OmniStack virtual machines. Finally you will need credentials with admin capabilities for using the OmniStack API. These credentials are typically the same as you vCenter admin credentials
L3 Network requirements	You will need one IP address for each and every VM configured in the Ansible inventory (see the section "Editing the inventory"). At the time of writing, the example inventory configures 12 virtual machines so you would need to allocate 12 IP addresses to use this example inventory. Note that the Ansible playbooks do not support DHCP so you need static IP addresses. All the IPs should be in the same subnet. You will also have to specify the size of the subnet (for example /22 or /24) and the L3 gateway for this subnet.
DNS	You will need to know the IP addresses of your DNS server. In addition, all the VMs you configure in the inventory should have their names registered in DNS. In addition, you will need the domain name to use for configuring the virtual machines (such as example.com)
NTP Services	You need time services configured in your environment. The solution being deployed (including Docker) uses certificates and certificates are time sensitive. You will need the IP addresses of your time servers (NTP).
RHEL Subscription	A RHEL subscription is required to pull extra packages that are not on the DVD.
Docker Prerequisites	You will need a URL for the official Docker EE software download and a license file. Refer to the Docker documentation to learn more about this URL and the licensing requirements at: https://docs.docker.com/engine/installation/linux/docker-ee/rhel/ in the section entitled "Docker EE repository URL"
Proxy	The playbooks pull the Docker packages from the Internet. If your environment accesses the Internet through a proxy, you will need the details of the proxy including the fully qualified domain name and the port number.

Enable vSphere High Availability

You must enable vSphere High Availability (HA) to support virtual machine failover during an HA event such as a host failure. Sufficient CPU and memory resources must be reserved across the system so that all VMs on the affected host(s) can fail over to remaining available hosts in the

system. You configure an Admission Control Policy (ACP) to specify the percentage CPU and memory to reserve on all the hosts in the cluster to support HA functionality.

More information on enabling vSphere HA and configuring Admission Control Policy is available in the HPE SimpliVity documentation. Log in to the HPE Support Center at https://www.hpe.com/us/en/support.html and search for "HPE SimpliVity 380". The administration guide is listed when you select the User document type.

Note

You should not use the default Admission Control Policy. Instead, you should calculate the memory and CPU requirements that are specific to your environment.

Install vSphere Docker Volume Service driver on all ESXi hosts

vSphere Docker Volume Service technology enables stateful containers to access the storage volumes provided by SimpliVity. This is a one-off manual step. In order to be able to use Docker volumes using the vSphere driver, you must first install the latest release of the vSphere Docker Volume Service (vDVS) driver, which is available as a vSphere Installation Bundle (VIB). To perform this operation, log in to each of the ESXi hosts, download and install the latest release of vDVS driver.

```
# esxcli software vib install -v /tmp/vmware-esx-vmdkops-<version>.vib --no-sig-check
```

More information on how to download and install the driver can be found on the Docker Store at https://store.docker.com/plugins/vsphere-docker-volume-service.

Note

You cannot mount the same persistent volume created through vSphere Docker Volume Service (vDVS) on containers running on two different hosts at the same time.

Create a VM template

The first step for the automated solution is the creation of a VM Template that you will use as the base for all your nodes. In order to create a VM Template you will first create a Virtual Machine with the OS installed and then convert the Virtual Machine to a VM Template. Since the goal of the automation is to remove as many repetitive tasks as possible, the VM Template is created as lean as possible, with any additional software installs and/or system configuration performed subsequently using Ansible.

It would be possible to automate the creation of the template. However, as this is a one-off task, it is appropriate to do it manually. The steps to create a VM template manually are described below:

1. Log in to vCenter and create a new Virtual Machine. In the dialog box, shown in Figure 3, select `Typical` and press `Next`:

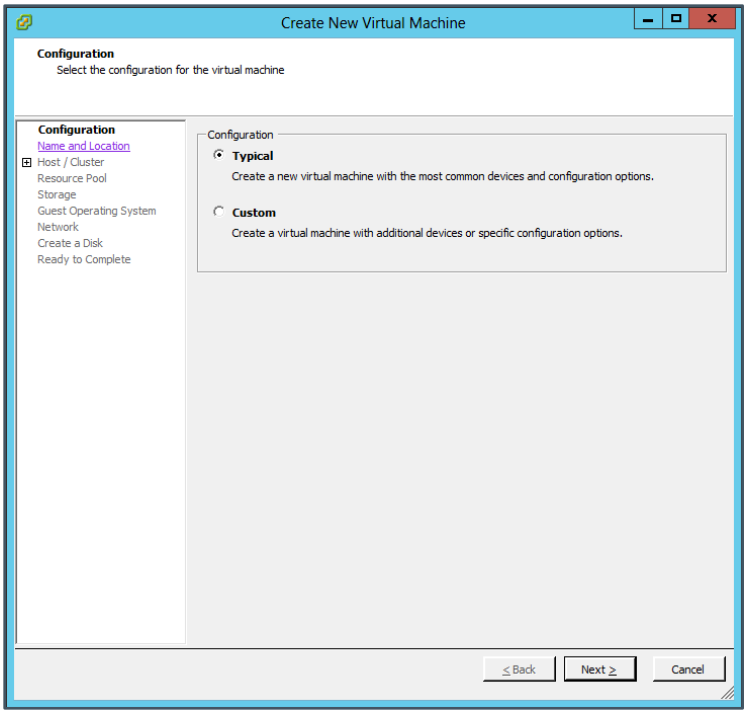

Figure 3. Create New Virtual Machine

2. Specify the name and location for your template, as shown in Figure 4:

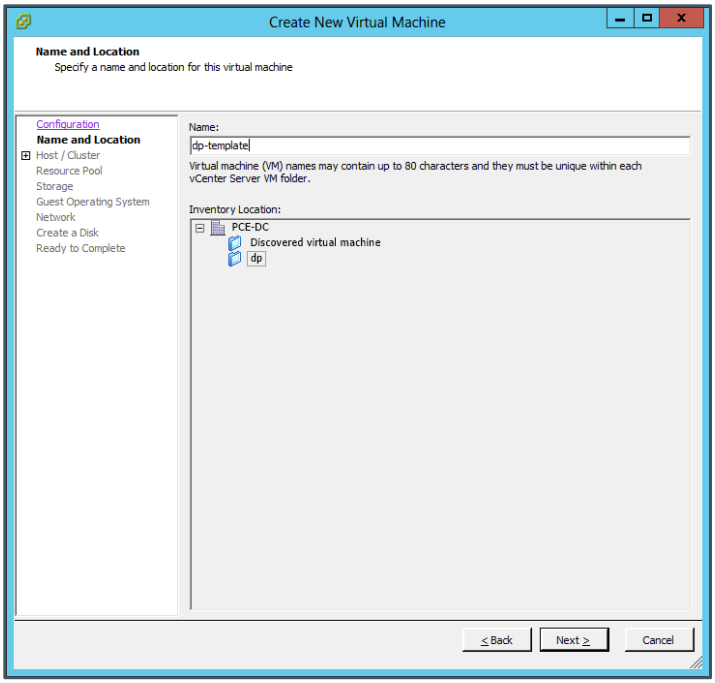

Figure 4. Specify name and location for the virtual machine

3. Choose the host/cluster on which you want to run this virtual machine, as shown in Figure 5.

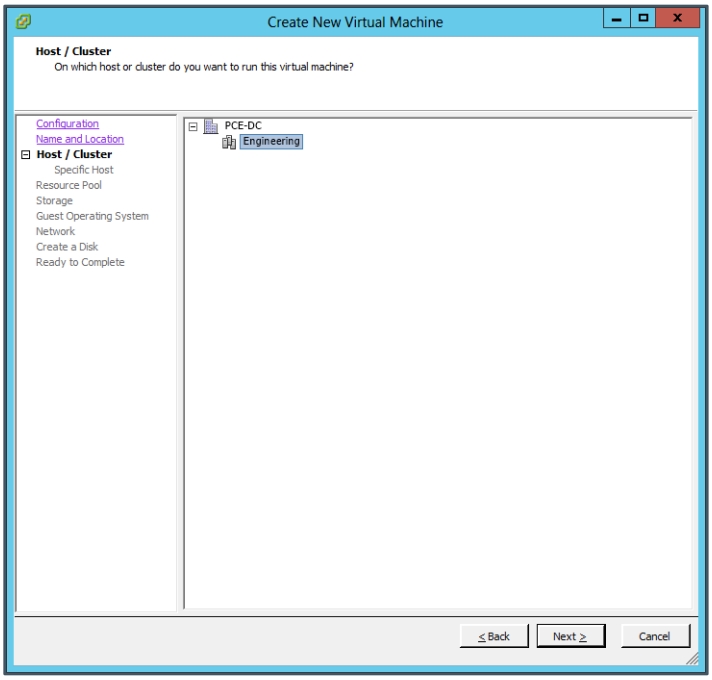

Figure 5. Choose host/cluster

4. Choose a datastore where the template files will be stored, as shown in Figure 6.

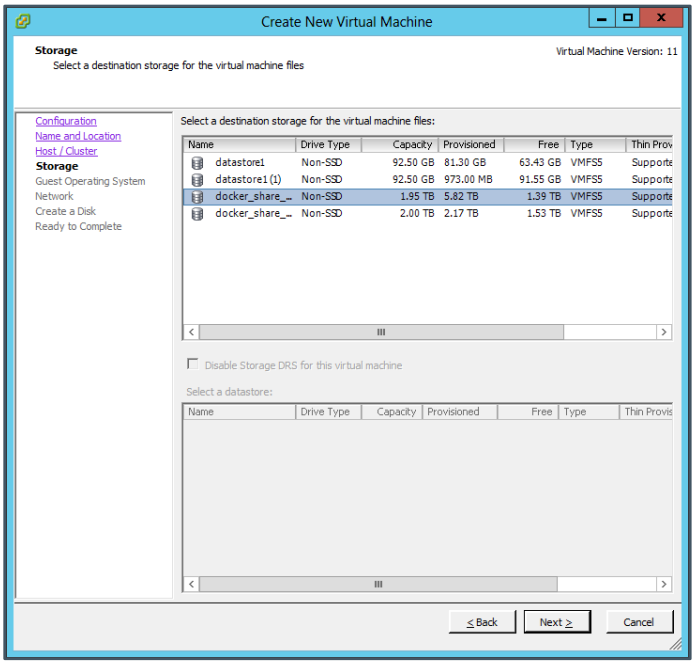

Figure 6. Select storage for template files

5. Choose the OS as shown in Figure 7, in this case Linux, RHEL 7 64bit.

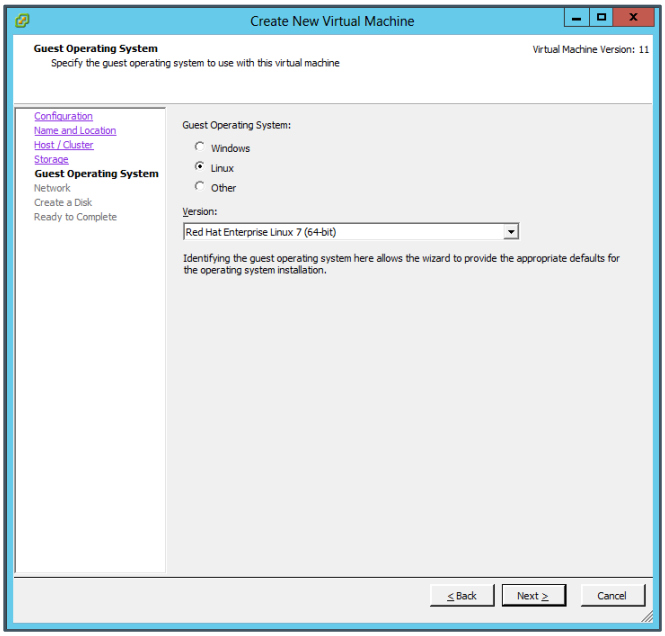

Figure 7. Choose operating system

6. Pick the network to attach to your template as shown in Figure 8. In this example there is only one NIC but depending on how you plan to architect your environment you might want to add more than one.

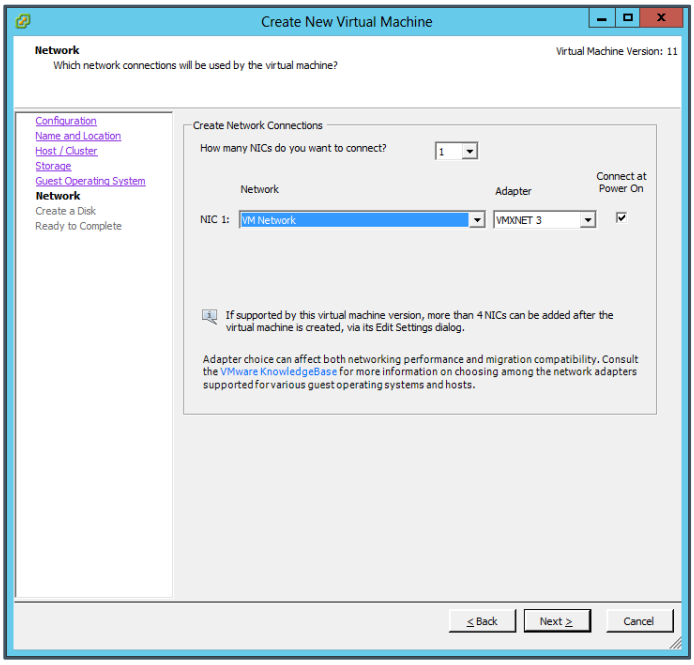

Figure 8. Create network connections

7. Create a primary disk as shown in Figure 9. The chosen size in this case is 50GB but 20GB should be typically enough. Select `Thin Provision` which does not pre-allocate space and press `Next`.

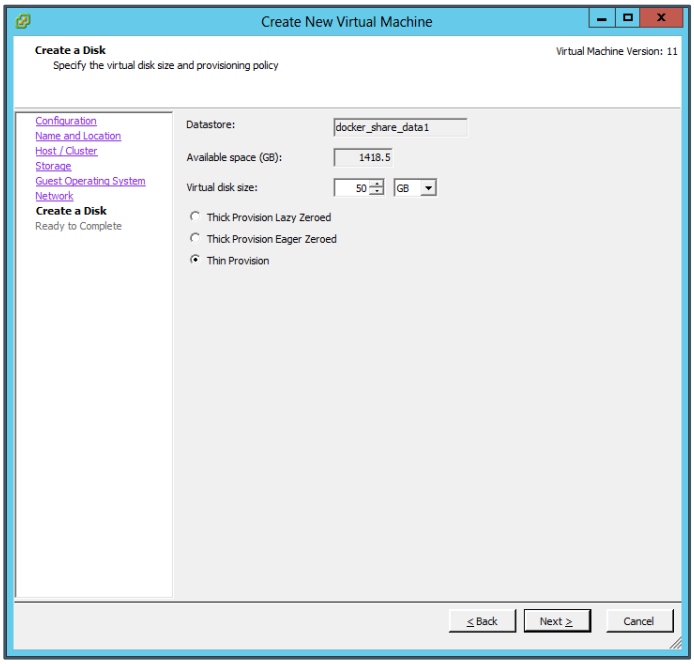

Figure 9. Create primary disk

8. Confirm that the settings are correct and press `Finish` as shown in Figure 10.

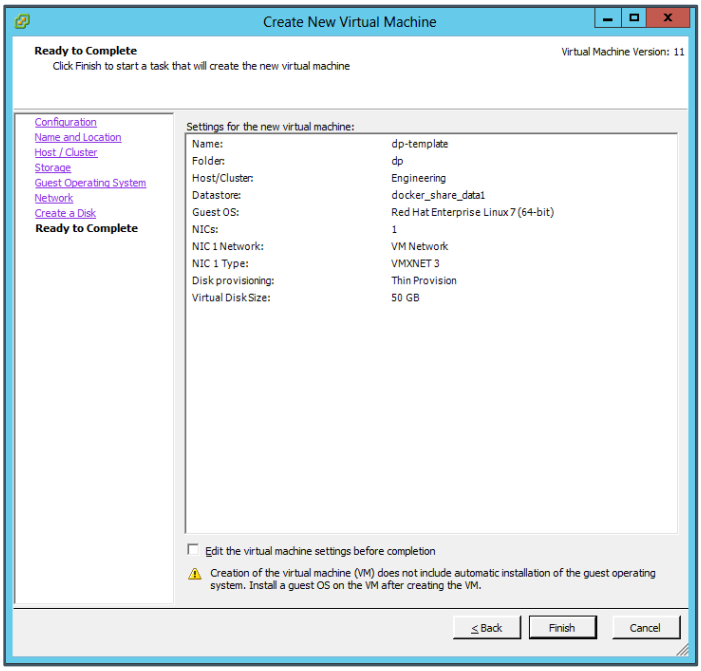

Figure 10. Confirm settings

9. The next step is to virtually insert the RHEL 7 DVD, using the Settings of the newly created VM as shown in Figure 11. Select your ISO file in the Datastore ISO File Device Type and make sure that the `Connect at power on` checkbox is checked.

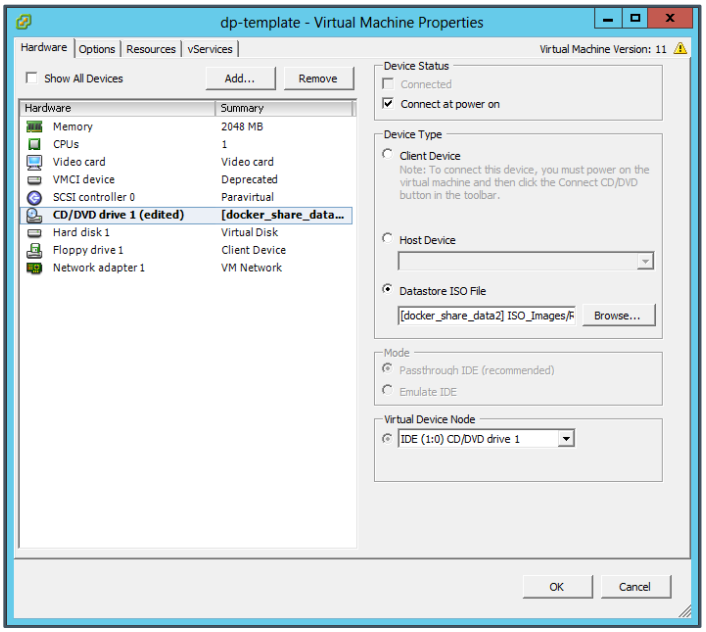

Figure 11. Virtual machine properties

10. Finally, you can optionally remove the Floppy Disk, as shown in Figure 12, as this is not required for the VM.

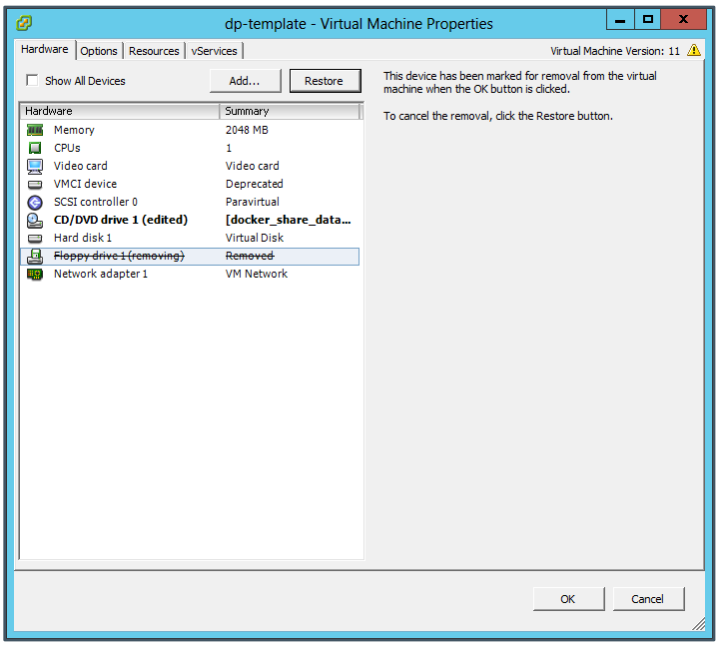

Figure 12. Remove Floppy drive

11. Power on the server and open the console to install the OS. On the welcome screen, as shown in Figure 13, pick your language and press `Continue`:

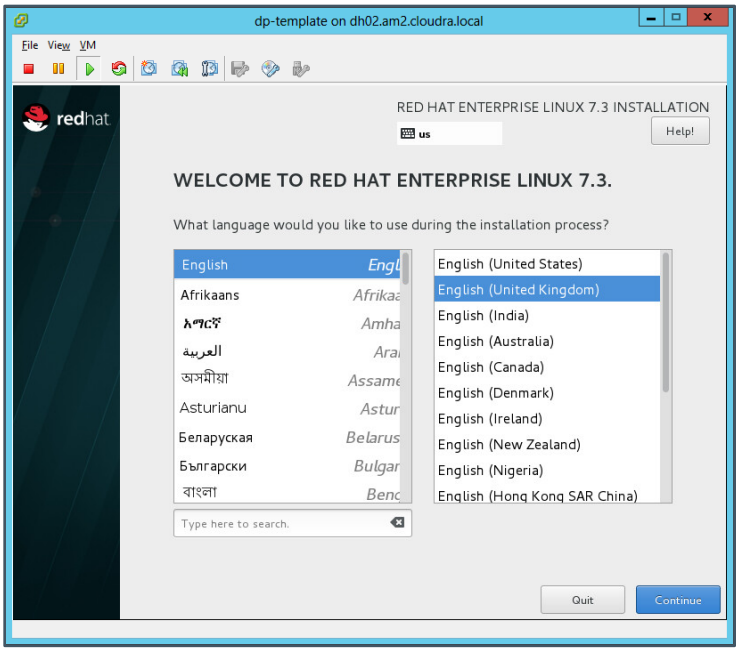

Figure 13. Welcome screen

12. The installation summary screen will appear, as shown in Figure 14.

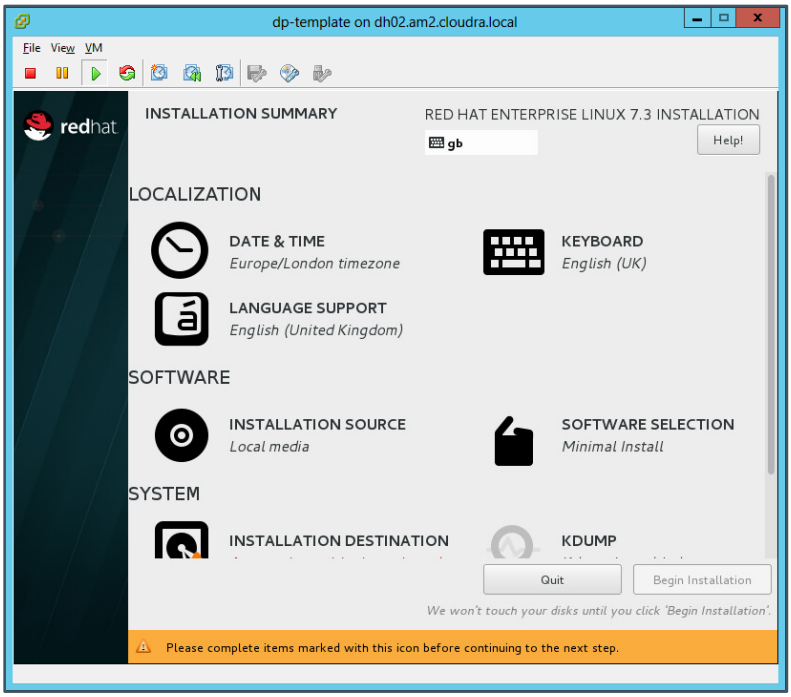

Figure 14. Installation summary

13. Scroll down and click on `Installation Destination`, as shown in Figure 15.

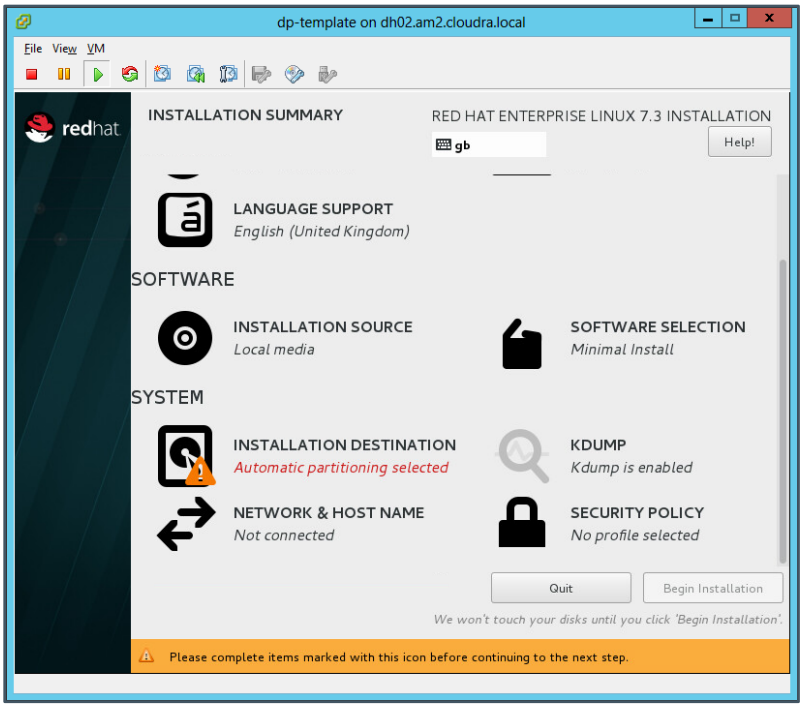

Figure 15. Installation destination

14. Select your installation drive, as shown in Figure 16, and click `Done`

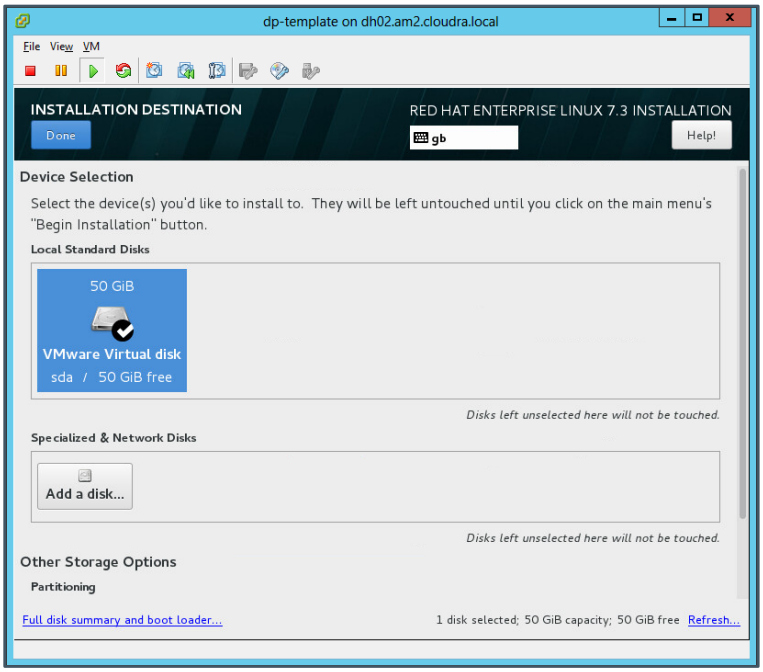

Figure 16. Select installation drive

15. Click `Begin Installation`, use all the other default settings, and wait for the Configuration of User Settings dialog, shown in Figure 17.

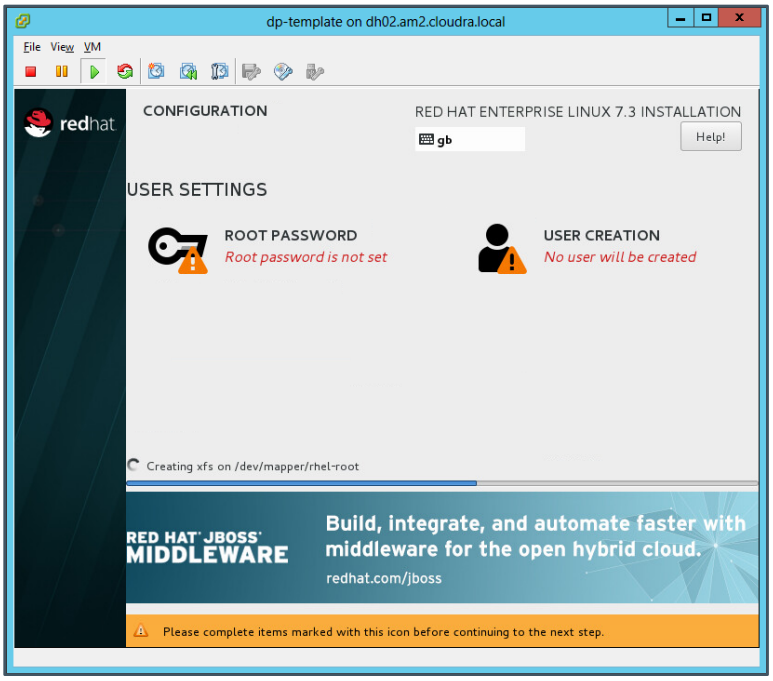

Figure 17. Configure user settings

16. Select a root password, as shown in Figure 18.

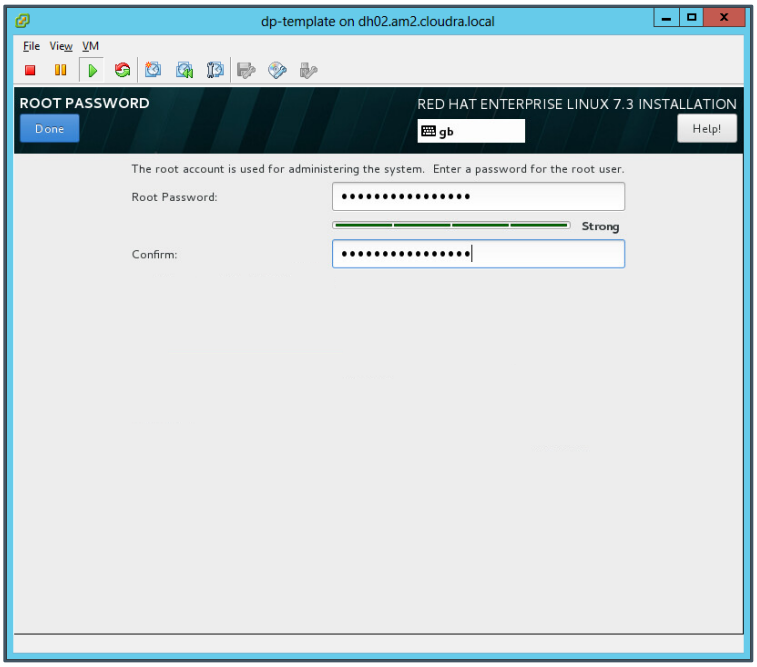

Figure 18. Set root password

17. Click Done and wait for the install to finish. Reboot and log into the system using the VM console.
18. The Red Hat packages required during the deployment of the solution come from two repositories: rhel-7-server-rpms and rhel-7-server-extras-rpms. The first repository can be found on the Red Hat DVD but the second cannot. There are two options, with both requiring a Red Hat Network account.
    a. Use Red Hat subscription manager to register your system. This is the easiest way and will automatically give you access to the official Red Hat repositories.

       If you are behind a proxy, you must configure the proxy before running command to register:

       ```
       # subscription-manager config --server.proxy_hostname=<proxy IP> --server.proxy_port=<proxy port>
       ```

       Use the subscription-manager register command as follows:

       ```
       # subscription-manager register --auto-attach
       ```

       If you use this option, the playbooks will automatically enable the extras repository on the VMs that need it. If you encounter proxy errors related to cdn.redhat.com, you should run the subscription-manager refresh command to recreate your certificate and then re-run the subscription-manager register command.

    b. Use an internal repository. Instead of pulling the packages from Red Hat, you can create copies of the required repositories on a dedicated node. You can then configure the package manager to pull the packages from the dedicated node. Your /etc/yum.repos.d/redhat.repo could look something like this:

       ```
       [RHEL7-Server]
       name=Red Hat Enterprise Linux $releasever - $basearch
       baseurl=http://yourserver.example.com/rhel-7-server-rpms/
       enabled=1
       gpgcheck=1
       gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release

       [RHEL7-Server-extras]
       name=Red Hat Enterprise Linux Extra pkg $releasever - $basearch
       baseurl=http://yourserver.example.com/rhel-7-server-extras-rpms/
       enabled=1
       gpgcheck=1
       gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
       ```

    The following articles explain how you can create a local mirror of the Red Hat repositories and how to share them:

    https://access.redhat.com/solutions/23016
    https://access.redhat.com/solutions/7227

Before converting the VM to a template, you will need to set up access for the Ansible host to configure the individual VMs. This is explained in the next section.

Create the Ansible node

In addition to the VM Template, you need another Virtual Machine where Ansible will be installed. This node will act as the driver to automate the provisioning of the environment and it is essential that it is properly installed. The steps are as follows:

1. Create a Virtual Machine and install your preferred OS (in this example, and for the sake of simplicity, RHEL 7 will be used). The rest of the instructions assume that, if you use a different OS, you understand the possible differences in syntax for the provided commands. If you use RHEL 7, select Infrastructure Server as the base environment and the Guests Agents add-on during the installation.
2. Log in the root account and create an SSH key pair. Do not protect the key with a passphrase (unless you want to use ssh-agent).

```
# ssh-keygen
```

3. Configure the following yum repositories, `rhel-7-server-rpms` and `rhel-7-server-extras-rpms` as explained in the previous section.
4. Configure the EPEL repository. For more information, see: http://fedoraproject.org/wiki/EPEL. Note that `yum-config-manager` comes with the Infrastructure Server base environment, if you did not select this environment you will have to install the `yum-utils` package.

   ```
   # rpm -ivh https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
   # yum-config-manager --enable rhel-7-server-extras-rpms
   ```

5. Install Ansible. The playbooks were tested with Ansible 2.2 and 2.3. Please note that the playbooks will not work with Ansible 2.4 due to an open defect https://github.com/ansible/ansible/issues/32000. Do not use Ansible 2.4 until this defect is fixed. To install the Ansible 2.3 use the following command:

   ```
   # yum install ansible-2.3.2.0-2.el7
   ```

6. Make a list of all the hostnames and IPs that will be in your system and update your `/etc/hosts` file accordingly. This includes your UCP node, DTR node, worker nodes, NFS server, logger server and load balancers.
7. Install the following packages which are a mandatory requirement for the playbooks to function as expected. (Update pip if requested).

   ```
   # yum install python-pyvmomi python-netaddr python2-jmespath python-pip gcc python-devel openssl-devel git
   # pip install --upgrade pip
   # pip install cryptography
   # pip install pysphere
   ```

8. Copy your SSH public key to the VM Template so that, in the future, your Ansible node can SSH without the need for a password to all the Virtual Machines created from the VM Template.

   ```
   # ssh-copy-id root@<VM_Template>
   ```

   Please note that, in both the Ansible node and the VM Template, you might need to configure the network so that one node can reach the other. Instructions for this step have been omitted since it is a basic step and will vary depending on your environment.

Finalize the template

Now that the VM Template has the public key of the Ansible node, you need to convert this VM to a VM Template. Perform the following steps in the VM Template to finalize its creation:

1. Clean up the template by running the following commands:

   ```
   # rm /etc/ssh/ssh_host_*
   # history -c
   ```

2. Shut down the VM:

   ```
   # shutdown -h now
   ```

3. Once the Virtual Machine is ready and turned off, it is time to convert it to a template as shown in Figure 19:

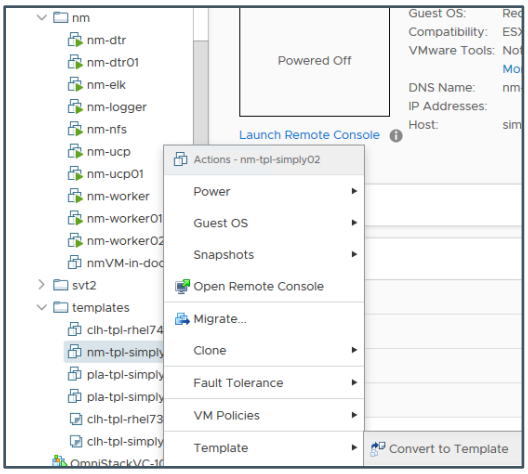

Figure 19. Convert to template

This completes the creation of the VM Template.

Prepare your Ansible configuration

On the Ansible node, retrieve the latest version of the playbooks using git.

```
# git clone https://github.com/HewlettPackard/Docker-SimpliVity
```

Change to the directory which you just cloned:
```
# cd ~/Docker-SimpliVity
```

Change to the `dev` directory:
```
# cd dev
```

Note

File names are relative to the `dev` directory. For example `vm_hosts` is located in `~/Docker-SimpliVity/dev` and `group_vars/vars` relates to `~/Docker-SimpliVity/dev/groups_vars/vars`.

You now need to prepare the configuration to match your own environment, prior to deploying Docker EE and the rest of the nodes. To do so, you will need to edit and modify three different files:

- `vm_hosts` (the inventory file)
- `group_vars/vars` (the group variables file)
- `group_vars/vault` (the encrypted group variable file)

You should work from the root account for the configuration steps and later when you run the playbooks.

Editing the inventory

The inventory is the file named `vm_hosts` in the `~Docker-SimpliVity/dev` directory. You need to edit this file to describe the configuration you want to deploy.

The nodes inside the inventory are organized in groups. The groups are defined by brackets and the group names are static so they must not be changed. All other information, including hostnames, specifications, IP addresses, etc., should be edited to match your requirements. The groups are as follows:

- [ucp_main]: A group containing one single node which will be the main UCP node and swarm leader. Do not add more than one node under this group.
- [ucp]: A group containing all the UCP nodes, including the main UCP node. For the Dev edition, you will only have the main node.
- [dtr_main]: A group containing one single node which will be the first DTR node to be installed. Do not add more than one node under this group.
- [dtr]: A group containing all the DTR nodes, including the main DTR node. For the Dev edition, you will only have the main node.
- [worker]: A group containing all the worker nodes. For the Dev edition, you will have two nodes under this group.
- [cloudbees]: A group containing the single node for CloudBees.
- [playwithdocker]: A group containing the single node for Play with Docker
- [ucp_lb]: A group containing one single node which will be the load balancer for the UCP node. Do not add more than one node under this group.
- [dtr_lb]: A group containing one single node which will be the load balancer for the DTR node. Do not add more than one node under this group.
- [worker_lb]: A group containing one single node which will be the load balancer for the worker nodes. Do not add more than one node under this group.
- [lbs]: A group containing all the load balancers. This group will have 3 nodes, also defined in the three groups above.
- [nfs]: A group containing one single node which will be the NFS node. Do not add more than one node under this group.
- [logger]: A group containing one single node which will be the logger node. Do not add more than one node under this group.
- [elk]: A group containing one single node which will be the ELK node. Do not add more than one node under this group.
- [local]: A group containing the local Ansible host. It contains an entry that should not be modified.

There are also a number of special groups:

- [docker:children]: A group of groups including all the nodes where Docker will be installed.
- [vms:children]: A group of groups including all the Virtual Machines involved apart from the local host.

Finally, you will find some variables defined for each group:

- [vms:vars]: A set of variables defined for all VMs. Currently only the size of the boot disk is defined here.
- [ucp:vars]: A set of variables defined for all nodes in the [ucp] group.
- [dtr:vars]: A set of variables defined for all nodes in the [dtr] group.
- [worker:vars]: A set of variables defined for all nodes in the [worker] group.
- [lbs:vars]: A set of variables defined for all nodes in the [lbs] group.
- [nfs:vars]: A set of variables defined for all nodes in the [nfs] group.
- [logger:vars]: A set of variables defined for all nodes in the [logger] group.
- [elk:vars]: A set of variables defined for all nodes in the [elk] group.

If you wish to configure your nodes with different specifications rather than the ones defined by the group, it is possible to declare the same variables at the node level, overriding the group value. For instance, you could have one of your workers with higher specifications by doing:

```
[worker]
worker01 ip_addr='10.0.0.10/16' esxi_host='esxi1.domain.local'
worker02 ip_addr='10.0.0.11/16' esxi_host='esxi1.domain.local' cpus='16' ram='32768'
[worker:vars]
cpus='4'
ram='16384'
disk2_size='200'
node_policy='bronze'
```

In the example above, the `worker02` node would have 4 times more CPU and double RAM compared to the other worker node.

The different variables you can use are described in Table 4 below. They are all mandatory unless specified otherwise.

Table 4. Variables

Variable	Scope	Description
`ip_addr`	Node	IP address in CIDR format to be given to a node.
`esxi_host`	Node	ESXi host where the node will be deployed. If the cluster is configured with DRS, this option will be overridden.
`cpus`	Node/Group	Number of CPUs to assign to a VM or a group of VMs.
`ram`	Node/Group	Amount of RAM in MB to assign to a VM or a group of VMs.
`disk2_usage`	Node/Group	Size of the second disk in GB to attach to a VM or a group of VMs. This variable is only mandatory on Docker nodes (UCP, DTR, worker) and NFS node. It is not required for the logger node or the load balancers.
`node_policy`	Node/Group	SimpliVity backup policy to assign to a VM or a group of VMs. The name has to match one of the backup policies defined in the `group_vars/vars` file described in the next section.

Editing the group variables

Once the inventory is ready, the next step is to modify the group variables to match your environment. To do so, you need to edit the file `group_vars/vars` under the cloned directory containing the playbooks. The variables can be defined in any order but for the sake of clarity they have been divided into sections.

VMware configuration

All VMware-related variables are mandatory and are described in Table 5.

Table 5. VMware variables

Variable	Description
`vcenter_hostname`	IP or hostname of the vCenter appliance
`vcenter_username`	Username to log in to the vCenter appliance. It might include a domain, for example, `administrator@vsphere.local`. **Note:** The corresponding password is stored in a separate file (`group_vars/vault`) with the variable named `vcenter_password`.
`vcenter_validate_certs`	'no'
`datacenter`	Name of the datacenter where the environment will be provisioned

vm_username	Username to log into the VMs. It needs to match the one from the VM Template, so unless you have created a user, you must use root. **Note:** The corresponding password is stored in a separate file (group_vars/vault) with the variable named vm_password.
vm_template	Name of the VM Template to be used. Note that this is the name from a vCenter perspective, not the hostname.
folder_name	vCenter folder to deploy the VMs. If you do not wish to deploy in a particular folder, the value should be /. **Note:** If you want to deploy in a specific folder, you need to create this folder in the inventory of the selected datacenter before starting the deployment.
datastores	List of datastores to be used, in list format, i.e. ['Datastore1','Datastore2'...]. Please note that from a SimpliVity perspective, it is a best practice to use only one Datastore. Using more than one will not provide any advantages in terms of reliability and will add additional complexity. This datastore must exist before you run the playbooks.
disk2	UNIX® name of the second disk for the Docker VMs. Typically /dev/sdb
disk2_part	UNIX name of the partition of the second disk for the Docker VMs. Typically /dev/sdb1
vsphere_plugin_version	Version of the vSphere plugin for Docker. The default is 0.19 which is the latest version at the time of writing this document. The version of the plugin should match the version of the vSphere Installation Bundle (VIB) that you installed on the ESXi servers.

HPE SimpliVity configuration

All HPE SimpliVity variables are mandatory and are described in Table 6.

Table 6. SimpliVity variables

Variable	Description
simplivity_username	Username to log in to the Simplivity Omnistack appliances. It might include a domain, for example, administrator@vsphere.local. **Note:** The corresponding password is stored in a separate file (group_vars/vault) with the variable named simplivity_password.
omnistack_ovc	List of Omnistack hosts to be used, in list format, i.e. ['omni1.local','onmi2.local'...]. If your OmniStack virtual machines do not have their names registered in DNS, you can use their IP addresses.
backup_policies	List of dictionaries containing the different backup policies to be used along with the scheduling information. Any number of backup policies can be created and they need to match the node_policy variables defined in the inventory. Times are indicated in minutes. All month calculations use a 30-day month. All year calculations use a 365-day year. The format is as follows: backup_policies: - name: daily' days: 'All' start_time: '11:30' frequency: '1440' retention: '10080' - name: 'hourly' days: 'All' start_time: '00:00'

	frequency: '60' retention: '2880'
dummy_vm_prefix	In order to be able to back up the Docker volumes, a number of "dummy" VMs need to be spin up. This variable will set a recognizable prefix for them
docker_volumes_policy	Backup policy to use for the Docker Volumes

Networking configuration

All network-related variables are mandatory and are described in Table 7.

Table 7. Network variables

Variable	Description
nic_name	Name of the device, for RHEL this is typically ens192 and it is recommended to leave it as is
gateway	IP address of the gateway to be used
dns	List of DNS servers to be used, in list format, i.e. ['8.8.8.8','4.4.4.4'...]
domain_name	Domain name for your Virtual Machines
ntp_server	List of NTP servers to be used, in list format, i.e. ['1.2.3.4','0.us.pool.net.org'...]

Docker configuration

All Docker-related variables are mandatory and are described in Table 8.

Table 8. Docker variables

Variable	Description
docker_ee_url	**Note:** This is a private link to your Docker EE subscription. This should be kept secret and defined in group_vars/vault. The value for docker_ee_url is the URL documented at the following URL:: https://docs.docker.com/engine/installation/linux/docker-ee/rhel/
rhel_version	Version of your RHEL OS, i.e: 7.3 The playbooks were tested with RHEL 7.3 and RHEL 7.4.
dtr_version	Version of the Docker DTR you wish to install. You can use a numeric version or latest for the most recent one. The playbooks were tested with 2.3.3 and 2.4.0.
ucp_version	Version of the Docker UCP you wish to install. You can use a numeric version or latest for the most recent one. The playbooks were tested with UCP 2.2.3 and 2.2.4.
images_folder	Directory in the NFS server that will be mounted in the DTR nodes and that will host your Docker images
license_file	Full path to your Docker license file (it should be stored in your Ansible host)
ucp_username	Username of the administrator user for UCP and DTR, typically admin. **Note:** The corresponding password is stored in a separate file (group_vars/vault) with the variable named ucp_password.
compose_vers	Used by Play with Docker. Default value is '1.16.1'
compose_url	The URL for downloading the correct version of Docker Compose. Depends on compose_vers.

Play with Docker configuration

Play with Docker (PWD) is a free, cloud-based service that gives users the ability to experiment and interact with real live Docker instances for testing or educational purposes. Variables associated with Play with Docker are described in Table 9.

Table 9. Play with Docker variables

Variable	Description
pwd_path	'/root/play-with-docker'
pwd_duration	'24h'

Environment configuration

All environment-related variables are described in **Table 10**.

Table 10. Environment variables

Variable	Description
env	Dictionary containing all environment variables. It contains three entries described below. Please leave the proxy related settings empty if not required: • http_proxy: HTTP proxy URL, for example, 'http://15.184.4.2:8080'. This variable defines the HTTP proxy URL if your environment is behind a proxy. • https_proxy: HTTP proxy URL, for example, 'http://15.184.4.2:8080'. This variable defines the HTTPS proxy URL if your environment is behind a proxy • no_proxy: List of hostnames or IPs that don't require proxy, for example, 'localhost,127.0.0.1,.cloudra.local,10.10.174.'

Editing the vault

Once your group variables file is ready, the next step is to create a vault file to match your environment. The vault file is similar to the group variables but it will contain all sensitive variables and will be encrypted.

There is a sample vault file named group_vars/vault.sample that you can use as a model for your vault file. To create a vault you create a new file group_vars/vault and add entries similar to:

```
---
vcenter_password: 'xxx'
docker_ee_url: 'yoururl'
vm_password: 'xxx'
simplivity_password: 'xxx'
ucp_password: 'xxx'
rhn_orgid: 'Red Hat Organization ID'
rhn_key: 'Red Hat Activation Key'
```

rhn_orgid and rhn_key are the credentials needed to subscribe the virtual machines with Red Hat Customer Portal. For more info regarding activation keys see the following URL: https://access.redhat.com/articles/1378093

To encrypt the vault you need to run the following command:

```
# ansible-vault encrypt group_vars/vault
```

You will be prompted for a password that will decrypt the vault when required. You can update the values in your vault by running:

```
# ansible-vault edit group_vars/vault
```

For Ansible to be able to read the vault, you need to specify a file where the password is stored, for instance in a file called `.vault_pass`. Once the file is created, take the following precautions to restrict access to this file:

1. Change the permissions so only `root` can read it using `chmod 600 .vault_pass`
2. Add the file to your `.gitignore` file if you are pushing the set of playbooks to a git repository

Running the playbooks

At this point, the system is ready to be deployed. Go to the root folder and run the following command:

```
# ansible-playbook -i vm_hosts site.yml --vault-password-file .vault_pass
```

The playbooks should run for 25-35 minutes depending on your server specifications and the size of your environment.

Post deployment

The playbooks are intended to be used to deploy a new environment. You should only use them for Day 0 deployment purposes.

Overview of the playbooks

Create virtual machines

The playbook `playbooks/create_vms.yml` will create all the necessary Virtual Machines for the environment from the VM Template defined in the vm_template variable.

Configure network settings

The playbook `playbooks/config_networking.yml` will configure the network settings in all the Virtual Machines.

Distribute public keys

The playbook `playbooks/distribute_keys.yml` distributes public keys between all nodes, to allow each node to password-less log in to every other node. As this is not essential and can be regarded as a security risk (a worker node probably should not be able to log in to a UCP node, for instance), this playbook is commented out in `site.yml` by default and must be explicitly uncommented to enable this functionality.

Register the VMs with Red Hat

The playbook `playbooks/config_subscription.yml` registers and subscribes all virtual machines to the Red Hat Customer Portal. This is only needed if you pull packages from Red Hat. This playbook is commented out by default but you should uncomment it to make sure each VM registers with the Red Hat portal. It is commented out so that you can test the deployment first without having to unregister all the VMs from the Red Hat Customer Portal between each test. If you are using an internal repository, as described in the section "Create a VM template", you can keep this playbook commented out.

Install HAProxy

The playbook `playbooks/install_haproxy.yml` installs and configures the HAProxy package in the load balancer nodes. HAProxy is the chosen tool to implement load balancing between UCP nodes, DTR nodes and worker nodes.

Install NTP

The playbook `playbooks/install_ntp.yml` installs and configures the NTP package in all Virtual Machines in order to have a synchronized clock across the environment. It will use the server or servers specified in the `ntp_servers` variable in the group variables file.

Install Docker Enterprise Edition

The playbook `playbooks/install_docker.yml` installs Docker along with all its dependencies.

Install rsyslog

The playbook `playbooks/install_rsyslog.yml` installs and configures rsyslog in the logger node and in all Docker nodes. The logger node will be configured to receive all syslogs on port 514 and the Docker nodes will be configured to send all logs (including container logs) to the logger node.

Configure Docker LVs

The playbook `playbooks/config_docker_lvs.yml` performs a set of operations on the Docker nodes in order to create a partition on the second disk and carry out the LVM configuration, required for a sound Docker installation.

Docker post-install configuration

The playbook `playbooks/docker_post_config.yml` performs a variety of tasks to complete the installation of the Docker environment.

Install NFS server

The playbook `playbooks/install_nfs_server.yml` installs and configures an NFS server on the NFS node.

Install NFS clients

The playbook `playbooks/install_nfs_clients.yml` installs the required packages on the DTR nodes to be able to mount an NFS share.

Install and configure Docker UCP nodes

The playbook `playbooks/install_ucp_nodes.yml` installs and configures the Docker UCP nodes defined in the inventory.

Install and configure DTR nodes

The playbook `playbooks/install_dtr_nodes.yml` installs and configures the Docker DTR nodes defined in the inventory. Note that serialization is set to 1 in this playbook as two concurrent installations of DTR may in some cases be assigned the same replica ID.

Install worker nodes

The playbook `playbooks/install_worker_nodes.yml` installs and configures the Docker Worker nodes defined in the inventory.

Install ELK stack

The playbook `playbooks/install_elk.yml` installs and configures the ELK stack and Logspout container defined in the inventory.

Install CloudBees Team Edition (Jenkins CI/CD Pipeline)

The playbook `playbooks/install_cloudbees.yml` installs CloudBees Team Edition software defined in the inventory.

Install Play with Docker

The playbook `playbooks/install_playwithdocker.yml` installs Play with Docker software defined in the inventory.

Configure dummy VMs to backup Docker volumes

The playbook `playbooks/config_dummy_vms_for_docker_volumes_backup.yml` ensures that you can backup Docker volumes that have been created using the vSphere plugin (vDVS) in SimpliVity. There is not a straight-forward way to do this, so you need to use a workaround. Since all Docker volumes are going to be stored in the `dockvols` folder in the datastore(s), you need to create a 'dummy' VM per datastore. The `vmx`, `vmsd` and `vmkd` files from this VM will have to be inside the `dockvols` folder, so when these VMs are backed up, the volumes are backed up as well. Obviously these VMs don't need to take any resources and you can keep them powered off.

Configure SimpliVity backups

The playbook `playbooks/config_simplivity_backups.yml` configures the defined backup policies in the group variables file in SimpliVity and will include all Docker nodes plus the 'dummy' VMs created before, so the existing Docker volumes are also taken in account. The playbook will mainly use the SimpliVity REST API to perform these tasks. A reference to the REST API can be found here: https://api.simplivity.com/rest-api_getting-started_overview/rest-api_getting-started_overview_rest-api-overview.html

Accessing the UCP UI

Once the playbooks have run and completed successfully, the Docker UCP UI should be available by browsing to the UCP load balancer or any of the nodes via HTTPS. The authentication screen will appear as shown in Figure 20:

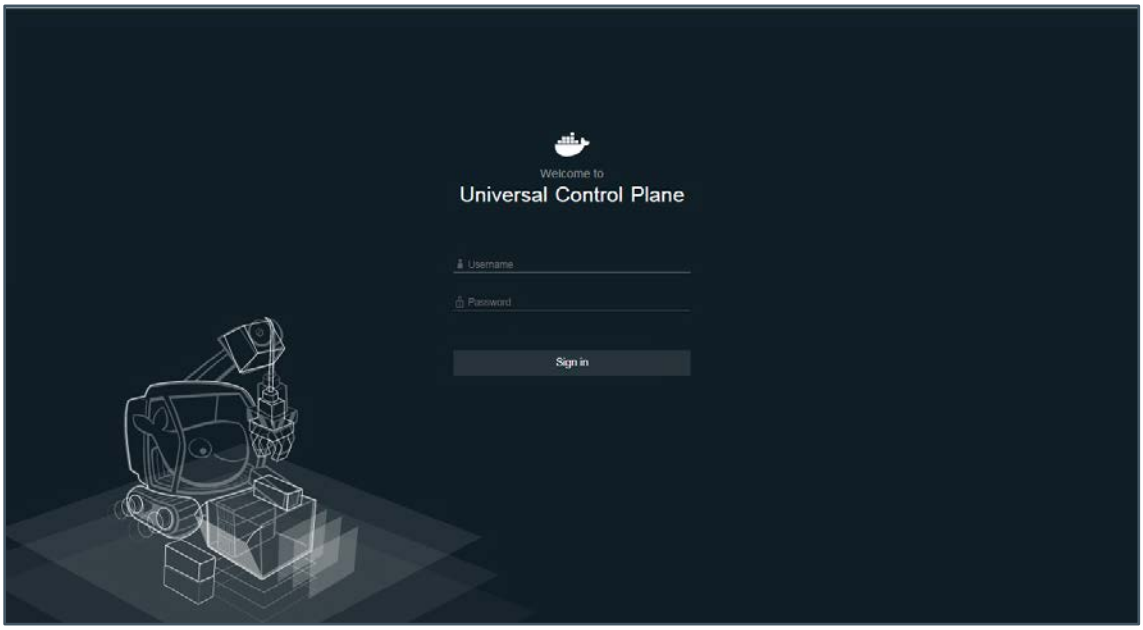

Figure 20. UCP authentication screen

Enter your credentials and the dashboard will be displayed as shown in Figure 21:

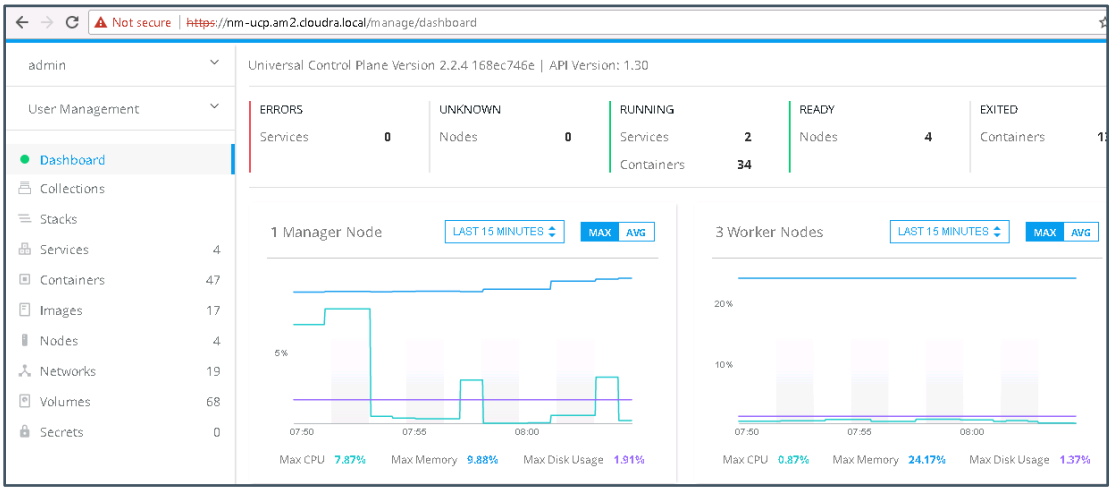

Figure 21. UCP dashboard

You should see all the nodes information in your Docker environment by clicking on Nodes, as shown in Figure 22:

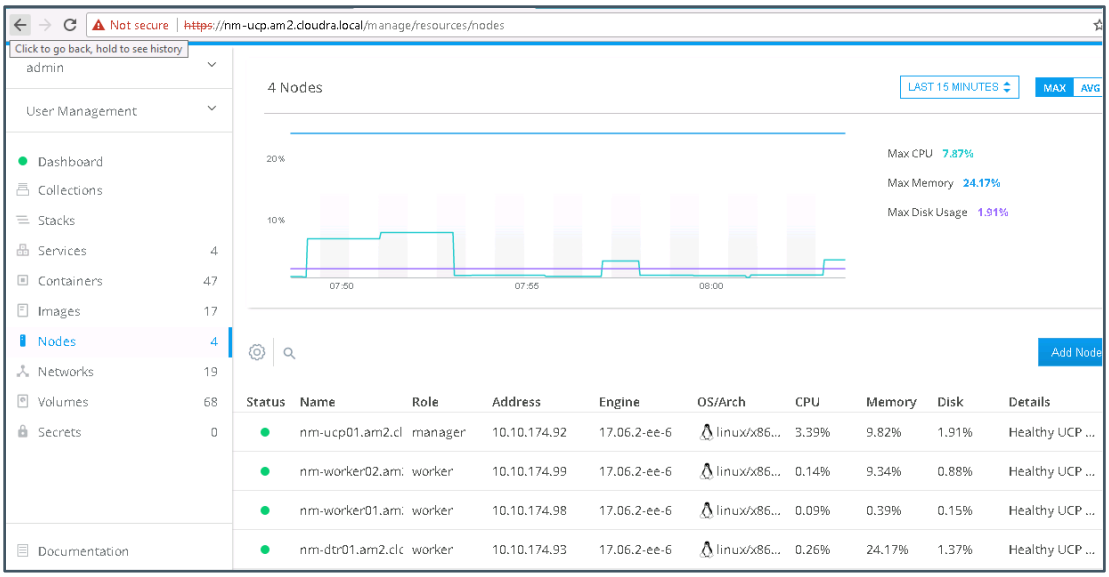

Figure 22. Nodes information

Click on `Services` to see the monitoring services that were installed during the playbooks execution, as shown in Figure 23:

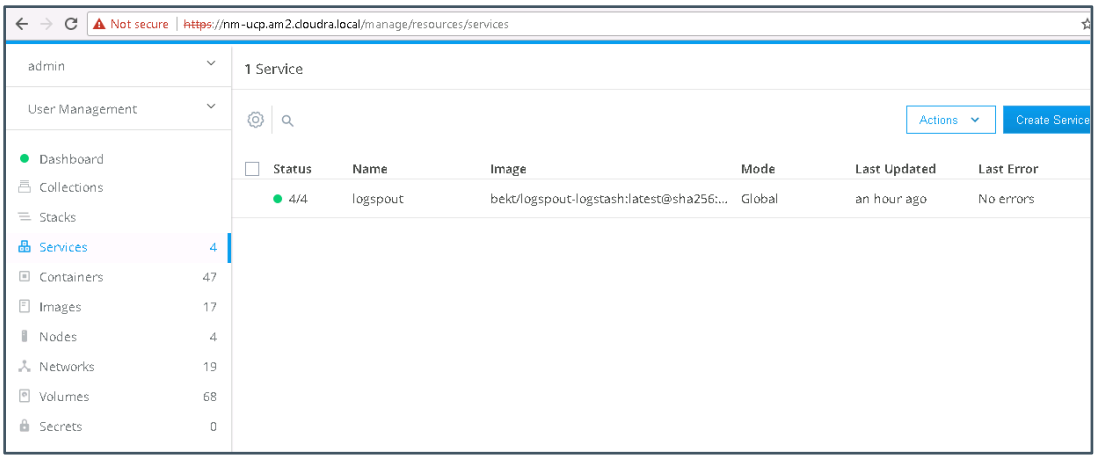

Figure 23. Services information

Accessing the DTR UI

The Docker DTR UI should be available by browsing to the DTR load balancer or any of the nodes via HTTPS. The authentication screen will appear as shown in Figure 24:

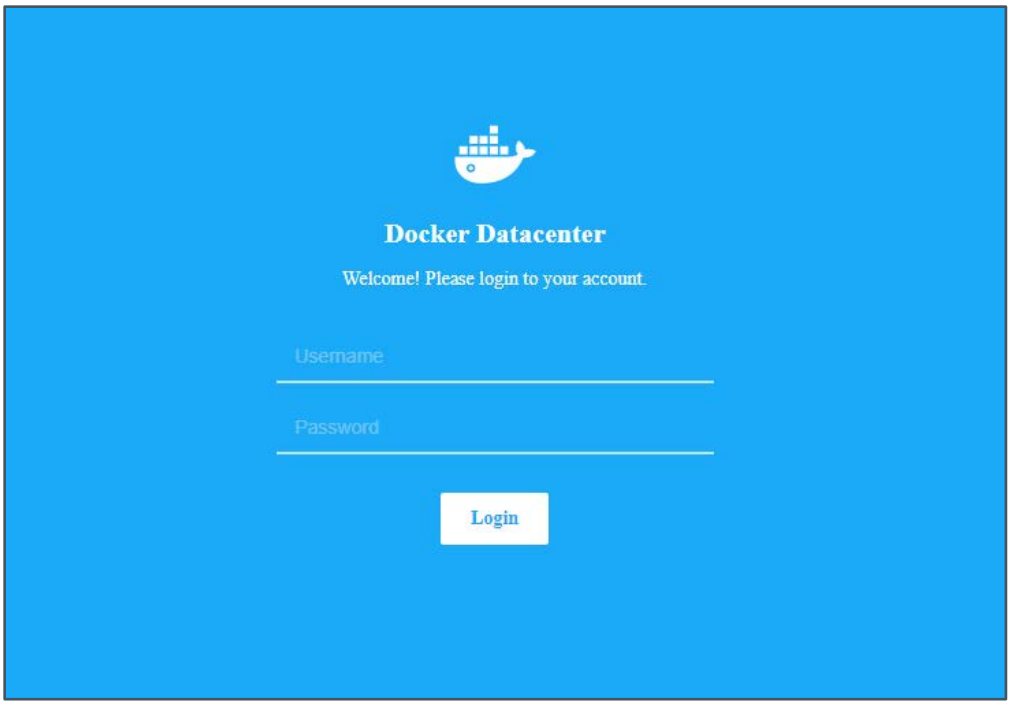

Figure 24. DTR authentication screen

Enter your UCP credentials and you should see the empty list of repositories as shown in Figure 25:

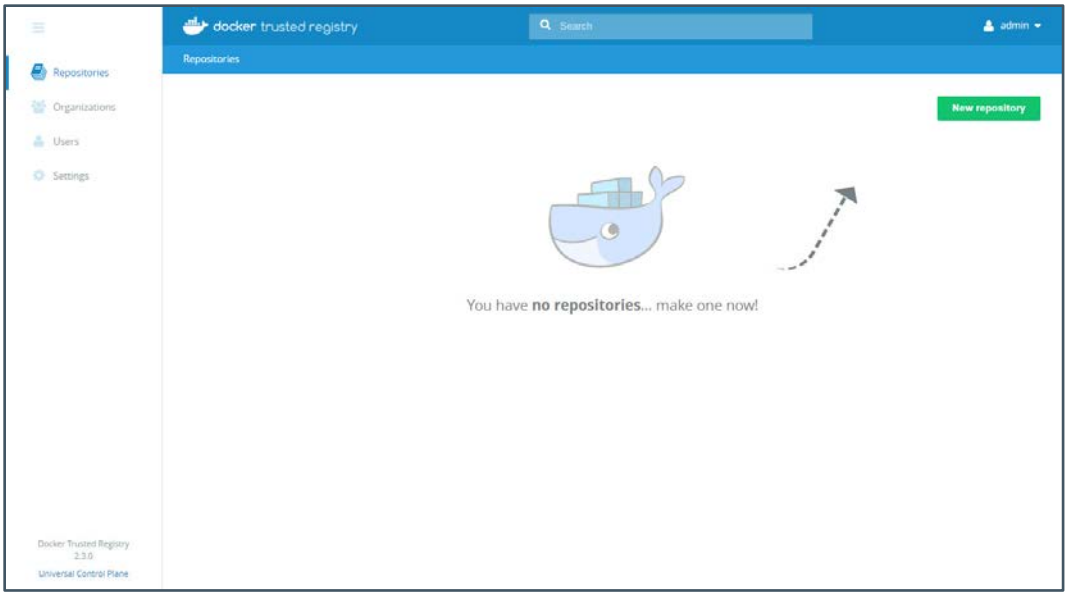

Figure 25. DTR repositories

If you navigate to Settings → Security, you should see the Image Scanning feature already enabled as shown in Figure 26. (Note that you need an Advanced license to have access to this feature).

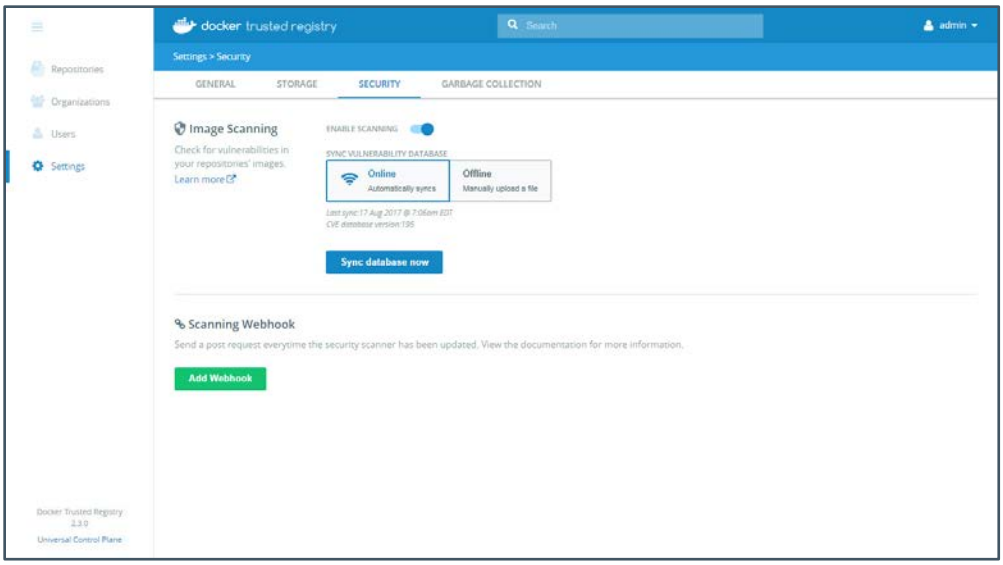

Figure 26. Image scanning in DTR

If you navigate to Settings → Storage, you should see that DTR is configured to use shared NFS storage as shown in Figure 27:

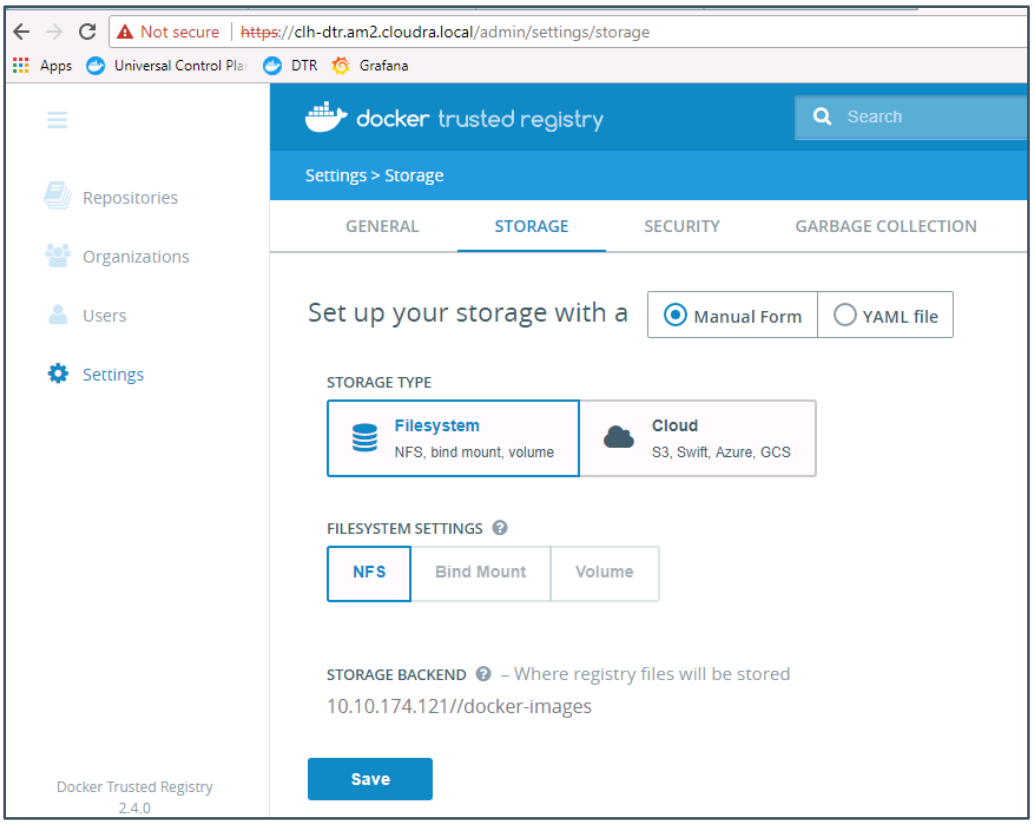

Figure 27. DTR storage settings

CloudBees Team Edition Solution

Instance Administration for CloudBees Jenkins Solutions

The CloudBees Jenkins Enterprise offering provides Managed Masters "as-a-service" on most public and private cloud providers, helping organizations to scale continuous integration and continuous delivery (CI / CD). The CloudBees Jenkins Team product, on the other hand, provides a single-instance option that allows small, independent teams to implement continuous integration and delivery.

CloudBees Jenkins Enterprise works best for large-scale installations while CloudBees Jenkins Team is designed for smaller-scale installations. Both products include verified integrations and worry-free upgrades that are enabled by and monitored through the Beekeeper Upgrade Assistant, and available through the CloudBees Assurance Program. You can see what plugins are available using the Beekeeper Upgrade Assistant, together with what ones have been updated and what ones you have already installed.

CloudBees Jenkins Team provides a stable Jenkins distribution, verified (or curated) plugins, and expert assistance. It can run as a stand-alone Java process installed manually for your operating system, or as a pre-configured Docker image, or within cloud-based infrastructure from Amazon (Amazon Web Service) or Microsoft (Azure). You can easily down-size from the Enterprise offering to the Team offering by removing the `license.xml` file in your JENKINS_HOME directory, disabling any unlicensed plugins, and restarting Jenkins.

This section is aimed at instance administrators who want to continuously build, test, and deliver applications using either Managed Masters or CloudBees Jenkins Team instance.

Building, testing, and deploying applications

A continuous delivery pipeline is an automated expression of your process for getting software from version control right through to your end users. Every change to your software (committed in source control) goes through a complex process on its way to being released. This process involves building the software in a reliable and repeatable manner, as well as the progression of the build through multiple stages of testing and deployment. Jenkins Pipeline is a suite of plugins that support implementing and integrating continuous delivery pipelines into Jenkins.

“Infrastructure as Code” is the process by which configuration and provisioning code is managed in the same way as source code. Typically, the definition of a Jenkins Pipeline is written into a text file (called a Jenkinsfile) which in turn is checked into a project’s source control repository. As a result, the continuous delivery pipeline is treated as a part of the application, to be versioned and reviewed like any other code.

Jenkins Pipeline provides an extensible set of tools for modeling simple-to-complex delivery pipelines, using a domain specific language (DSL) syntax based on the Groovy programming language. The Pipeline code can be entered directly using the Jenkins UI, but for more complex processes, it is more likely that you will use a script that is loaded from your source control management (SCM) system.

Accessing CloudBees Jenkins Team

Details for configuring the CloudBees Jenkins Team Edition installation are available in Appendix B. Once the configuration is complete, you can access the UI by browsing to `http://worknode02:8080` as shown in Figure 28.

Figure 28. Jenkins UI

Container logs monitoring using ELK monitoring

ELK Stack

ELK is a framework used to collect, filter and visualize log data. The ELK stack is a set of open source tools developed by Elastic.co and consists of:

- **E**lasticsearch, a database to store logs
- **L**ogstash, a data collection engine with pipelining capabilities.
- **K**ibana, a front end to parse, filter and visualize logs

The data flow in the stack is shown in **Figure 29**.

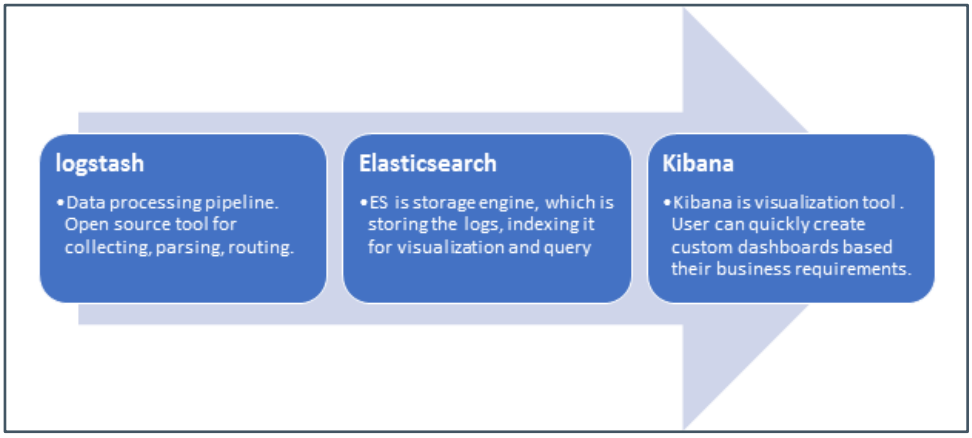

Figure 29. ELK data flow

Log aggregation

Each Docker Container Engine is configured to send its logs to a common aggregation point. Tools can then be used to process the logs, deploying filters, searches and dashboards. The logs from the containers on each worker node are captured using a `logspout` container running on the node and are then routed to `logstash`, running on ELK node. Figure 30 depicts the logical layout of the monitoring architecture.

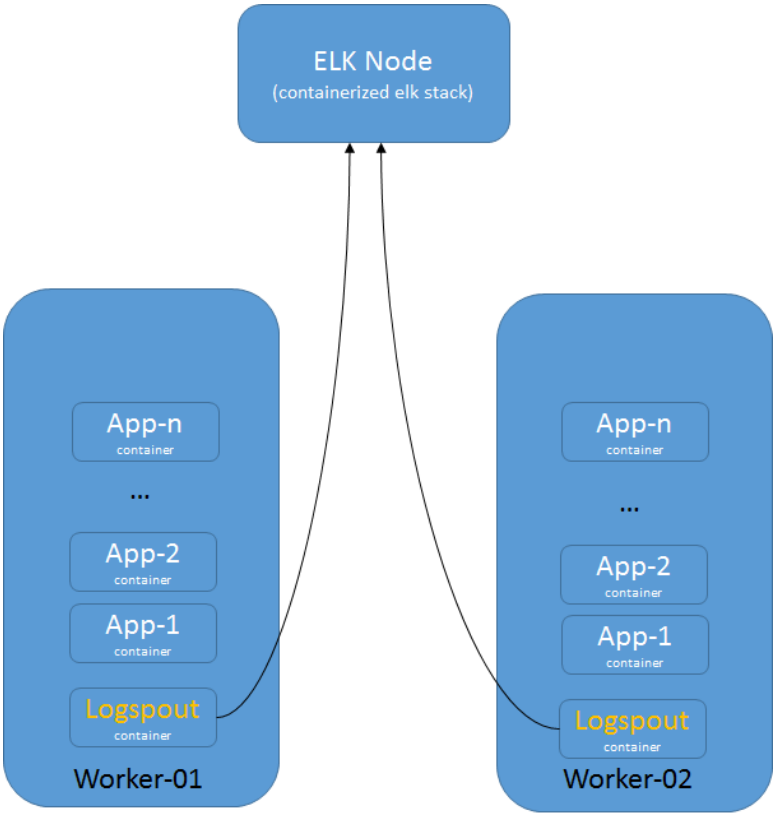

Figure 30. Monitoring architecture

Data flow

Logspout routes the data from all the containers running on each of the worker nodes to Logstash running on port `5000` on the ELK node. You can configure Logspout to filter the data based on business rules that you define in the `files/logstash.conf` file. Logstash also allows the user to define the destination or "stash" for the data. In the current configuration, Elasticsearch has been defined for storage. For more details on how to filter and transform the data on the fly, refer to the documentation at https://www.elastic.co/products/logstash.

Data visualization

Kibana is used for data visualization. Kibana queries Elasticsearch periodically and helps a user to create a monitoring dashboard. For more information on configuring Kibana, see Appendix C.

Deploying the Play with Docker service

Play with Docker (PWD) is a free, cloud-based service that gives users the ability to experiment and interact with real live Docker instances for testing or educational purposes. PWD provides the experience of deploying an Alpine Linux Virtual Machine, where you can build and run Docker containers and even create clusters with Docker features like Swarm Mode. The service allows users who have no previous Docker experience or no access to local hardware with the opportunity to work in a live Docker environment.

Under the hood, Docker-in-Docker (DinD) is used to give the effect of multiple VMs / PCs. In addition to the playground, PWD also includes a training site composed of a large set of Docker labs and quizzes from beginner to advanced level available at http://training.play-with-docker.com.

The public cloud-based version of PWD is available at http://play-with-docker.com. The service allows users to create up to 5 Docker nodes per session. Users have the option of deploying a 5-node Docker Swarm, a 3-node Swarm with 2 worker nodes, or 5 separate worker nodes. http://play-with-docker.com. The service allows users to create up to 5 Docker nodes per session. Users have the option of deploying a 5-node Docker Swarm, a 3-node Swarm with 2 worker nodes, or 5 separate worker nodes.

While the cloud-based version of PWD is great for learning and experimentation, enterprise users interested in testing their proprietary application code in a Docker environment may be hesitant to upload their source code to an external cloud-based environment due to security concerns. In addition, the cloud-based PWD service has a strict time limit, allowing users to work with their Docker instances for up to 4 hours. After this time, the entire environment is automatically deleted.

For these reasons, HPE is providing a locally deployed Play with Docker service as part of this solution stack. Additionally, a slightly modified version of the PWD service is deployed, allowing users to extend the session timer to as long as 24 hours. This locally-deployed instance with a customizable session timer allows enterprise customers, interested in quickly and painlessly testing their applications in a Docker environment, a much safer and more secure experience.

Before you can use Play with Docker (PWD), you must first decide on what software package to use to expose your desktop display environment. VNC or the main Server GUI are popular choices, as your UCP node doesn't have a GUI installed as a default. PWD requires a node with manager role permissions so you should use the main UCP node. This example uses the "Server with GUI" packages, but feel free to use VNC or your own preferred method.

Connect using `ssh` to main UCP node which is running RHEL 7, and run the following commands to install the packages to enable access to a graphic console desktop.

```
# yum -y groupinstall "Server with GUI"
# init 5 "This will alter your current run level from 3 to 5"
# systemctl set-default graphical.target "Sets the default run level to graphical"
```

Now, from your vSphere client, open the console for the UCP node and carry out the initial setup steps. Select the language and time zone and then create a Play with Docker user account. Once the desktop is visible, click on Applications, select the Firefox web browser and browse to `http://localhost`.

For more information on configuring Play with Docker, see Appendix D.

Security considerations

In addition to having all logs centralized in a single place and the image scanning feature enabled for the DTR nodes, there are other guidelines that should be followed in order to keep your Docker environment as secure as possible. The HPE Reference Configuration paper for securing Docker on HPE Hardware places a special emphasis on securing Docker in DevOps environments and covers best practices in terms of Docker security. The document can be found at: http://h20195.www2.hpe.com/V2/GetDocument.aspx?docname=a00020437enw. Some newer Docker security features that were not covered in the reference configuration are outlined below.

Prevent tags from being overwritten

By default, an image tag can be overwritten by a user with the correct access rights. As an example, an image such as `library/wordpress:latest` can be pushed and tagged with by different users, yet have different functionality. This might make it difficult to trace back the image to the build that generated it.

Docker DTR can prevent this from happening with the immutable tags feature that can be configured on a per repository basis. Once an image is pushed with a tag, that particular tag cannot be overwritten.

More information about immutable tags can be found at:
https://docs.docker.com/datacenter/dtr/2.3/guides/user/manage-images/prevent-tags-from-being-overwritten/

Isolate swarm nodes to a specific team

With Docker EE Advanced, you can enable physical isolation of resources by organizing nodes into collections and granting Scheduler access for different users. To control access to nodes, move them to dedicated collections where you can grant access to specific users, teams, and organizations.

More information about this subject can be found at: https://docs.docker.com/datacenter/ucp/2.2/guides/access-control/isolate-volumes-between-teams/.

Solution lifecycle management

Introduction

Lifecycle management with respect to this solution refers to the maintenance and management of software and hardware of various components that make up the solution stack. Lifecycle management is required to keep the solution up-to-date and ensure that the latest versions of the software are running to provide optimal performance, security and fix any existing defects within the product.

In this section, we will cover lifecycle management of the different components that are used in this solution. The architectural diagram of the solution in Figure 31 shows the software and hardware stacks that make up the solution.

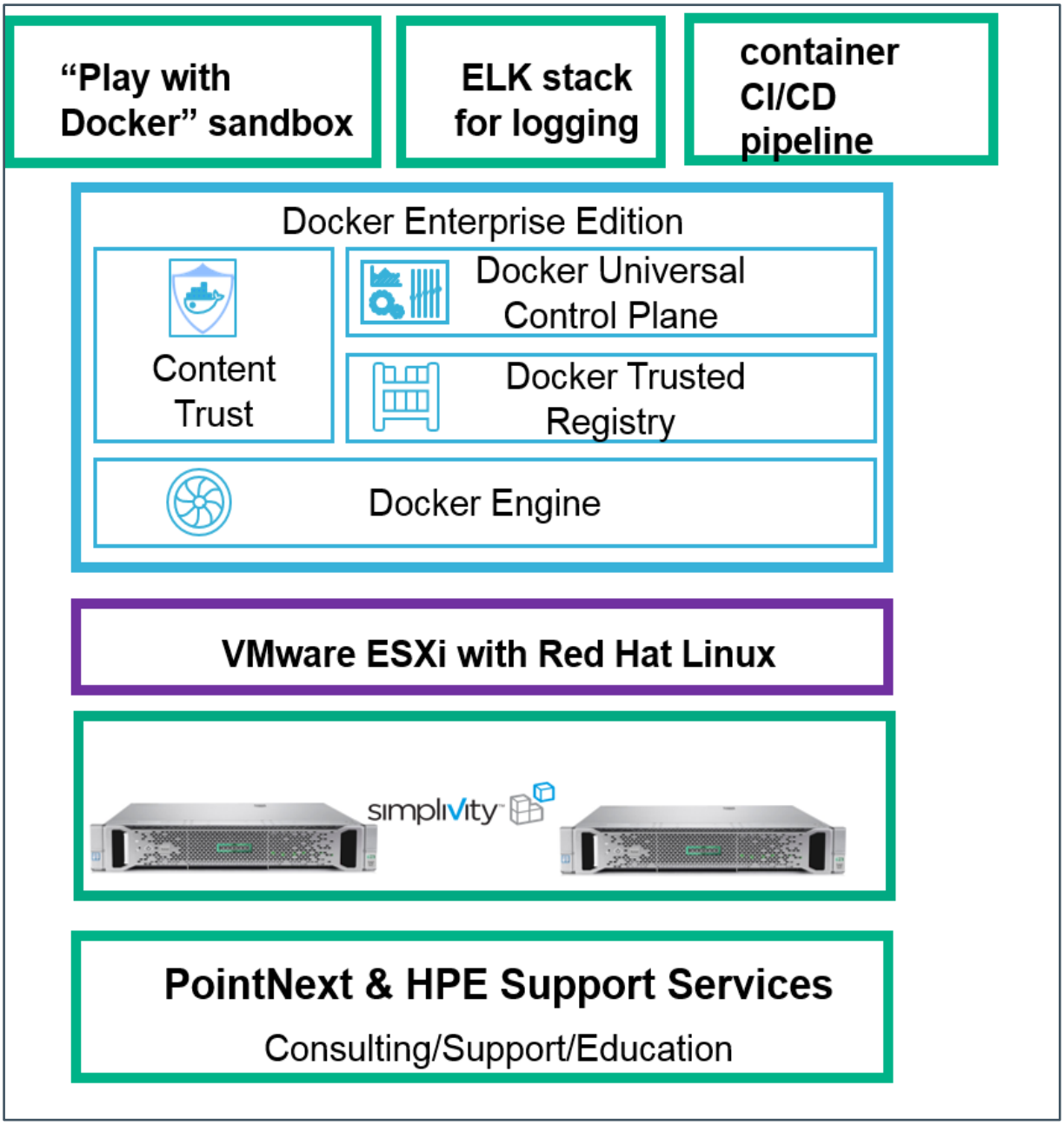

Figure 31. Solution stack

HPE SimpliVity environment

The HPE SimpliVity environment is made up of proprietary SimpliVity software, VMware software and HPE firmware. There are interdependencies between the various components that need to be accounted for and are provided in the table below. The components in Table 11 are part of the SimpliVity environment that require lifecycle management.

In general, ensure that the software bits for the Arbiter and vSphere extension corresponding to an OmniStack release are used.

Table 11. SimpliVity components

Order	Component	Dependency (compatibility)	Download/Documentation
1	HPE SimpliVity Arbiter	1. HPE OmniStack	SimpliVity OmniStack for vSphere Upgrade Guide Download software bits from HPE's support website. http://www.hpe.com/support
2	HPE SimpliVity VMware Plug-in	1. HPE SimpliVity Arbiter 2. HPE OmniStack	
3	HPE Omnistack	1. HPE SimpliVity VMware Plug-in 2. HPE SimpliVity Arbiter	

VMware components

The SimpliVity solution used in this deployment guide is built on VMware vSphere. VMware ESXi and vCenter (see Table 12) are the two components from VMware that are leveraged by the SimpliVity software.

The VMware ESXi and vCenter versions must be compatible with each other and with the HPE OmniStack version that is running on the SimpliVity systems.

Table 12. VMware components

Order	Component	Dependency (compatibility)	Download/Documentation
1	VMware vCenter	1. HPE OmniStack 2. VMware ESXi	VMware Upgrade for SimpliVity
2	VMware ESXi	1. HPE OmniStack 2. VMware vCenter	

HPE server software

SimpliVity servers are based on HPE server platforms and require compatible firmware version to function with HPE OmniStack Software, as shown in Table 13.

Table 13. HPE server components

Order	Component	Dependency (compatibility)	Download/Documentation
1	HPE Firmware	1. HPE OmniStack Software	Firmware Upgrade for SimpliVity

vSphere Docker Volume Service plug-in

vSphere Docker Volume Service plug-in is part of an open source project by VMware that enables running stateful containers by providing persistent Docker volumes leveraging existing storage technology from VMware. There are two parts to the plug-in, namely, client software and

server software (see Table 14). Every version of the plug-in that is released includes both pieces of software and it is imperative that the version number installed on the client side and server side are the same.

When updating the Docker Volume Service plug-in, ensure the ESXi version you are running is supported and that the client software is compatible with the operating system.

Table 14. vSphere Docker Volume Service components

Order	Component	Dependency (compatibility)	Download/Documentation
1	Server Software	1. VMware ESXi 2. Docker EE	vSphere Docker Volume Service on GitHub
2	Client Software	1. VM Operating System 2. Docker EE	

Red Hat Enterprise Linux operating system

This solution is built using Red Hat Enterprise Linux (see Table 15) as the base operating system. When upgrading the operating system on the VMs, first verify that the OS version is compatible to run Docker EE by looking at the Docker OS compatibility metric.

Table 15. Operating system

Order	Component	Dependency	Download/Documentation
1	Red Hat Enterprise Linux	1. Docker EE 2. vDVS client software plugin	RHEL

Docker EE environment

Each release of Docker Enterprise Edition contains three technology components – UCP, DTR and the Docker Daemon or Container Engine. It is imperative that the components belonging to the same version are deployed or upgraded together – see Table 16.

A banner will be displayed on the UI, as shown in Figure 32, when an update is available for UCP or DTR. You can start the upgrade process by clicking on the banner.

Figure 32. Docker update notification

Table 16. Docker EE components

Order	Component	Dependency	Download/Documentation
1	Docker Daemon/Engine	1. VM Operating System 2. vDVS plugin	1. Docker Lifecycle Maintenance 2. Docker Compatibility Matrix
2	Universal Control Plane		

Appendix A: Bill of Materials

This is a recommended BOM for the 2 nodes HPE Express Containers with Docker EE: Dev Edition. Please verify with your HPE account team to ensure these are the latest BOM SKUs.

Table 17. BOM for each SimpliVity node. Dual Socket – 14 cores per Socket, 5x1.92TB value flash

Product #	Product Description	Qty
Q8D81A	HPE SimpliVity 380 Gen10 Node	1
826856-B21	2.2GHz Xeon Gold 5120 processor (1 chip, 14 cores)	1
826856-L21	2.2GHz Xeon Gold 5120 processor (1 chip, 14 cores)	1
Q8D85A	HPE SimpliVity 240G 12 DIMM FIO Kit	2
Q8D91A	HPE SimpliVity 380 SM Val Kit	1
804331-B21	Smart Array P408i-a SR Gen10 (8 Internal Lanes/2GB Cache) 12G SAS Modular Controller	1
826703-B21	HPE DL380 Gen10 Sys Insght Dsply Kit	1
665243-B21	HPE Ethernet 10Gb 2P 560FLR-SFP+ Adptr (SFP+ connector)	1
864279-B21	HPE TPM 2.0 Gen10 Kit	1
865414-B21	800W Flex Slot Platinum Hot Plug Low Halogen Power Supply Kit	2
733664-B21	HPE 2U Cable Management Arm for Easy Install Rail Kit	1
867809-B21	HPE Gen10 2U Bezel Kit	1
758959-B22	HPE Legacy FIO Mode Setting	1
874543-B21	HPE 1U Gen10 SFF Easy Install Rail Kit	1
	BD505A - HPE iLO Adv incl 3yr TSU	
BD505A	1-Svr Lic (HPE)	1
Q8A60A	HPE OmniStack 8-14c 2P Small SW	1

Appendix B: CloudBees Team Edition configuration

The playbook `install_cloudbees.yml` installs the CloudBees Team Edition software. To configure the setup, you need to browse to `http://worknode02:8080` and login for the first time. The initial `admin` password is available from the container running CloudBees, in the file `/var/jenkins_home/secrets/initialAdminPassword`. To retrieve the password, determine the container ID using the `docker ps` command and then execute a `bash` command on that container to display the password.

```
# docker ps | grep cloudbees-jenkins-team
CONTAINER ID    IMAGE …
354d8d65242e    cloudbees/cloudbees-jenkins-team …

# docker exec -it 354d8d65242e bash
/ # cat /var/Jenkins_home/secrets/initialAdminPassword
754ba6ca54904c43bc04ea5d0dc74298
```

After you login as the `admin` user, you can either enter a valid license key or request a trial license, as shown in Figure 33.

Figure 33. License options

To obtain a license, you need to open a support ticket with CloudBees, specifying your instance ID which is displayed when you click either option in the dialog. For more information on license generation, see the documentation available at https://support.cloudbees.com/hc/en-us/articles/218233918-Jenkins-Enterprise-License-Activation.

Once you enter a valid (or trial) license key, you can proceed to customize Jenkins. It is recommended that you install the suggested plugins, as shown in Figure 34, unless you have a specific requirement for a particular plugin or plugins to be installed.

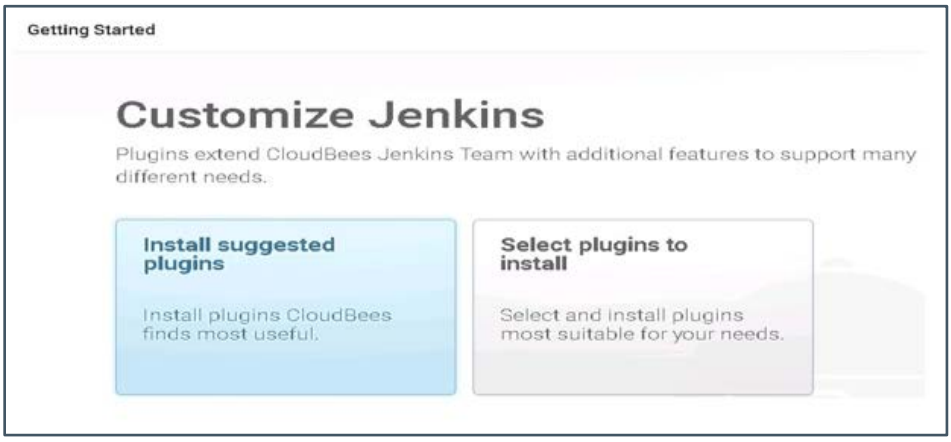

Figure 34. Customize Jenkins

As the plugins are installed, they will change to a green color in the user interface, as shown in Figure 35. The text box on the right hand side of the screen provides details of the plugins and any dependencies that need to be installed.

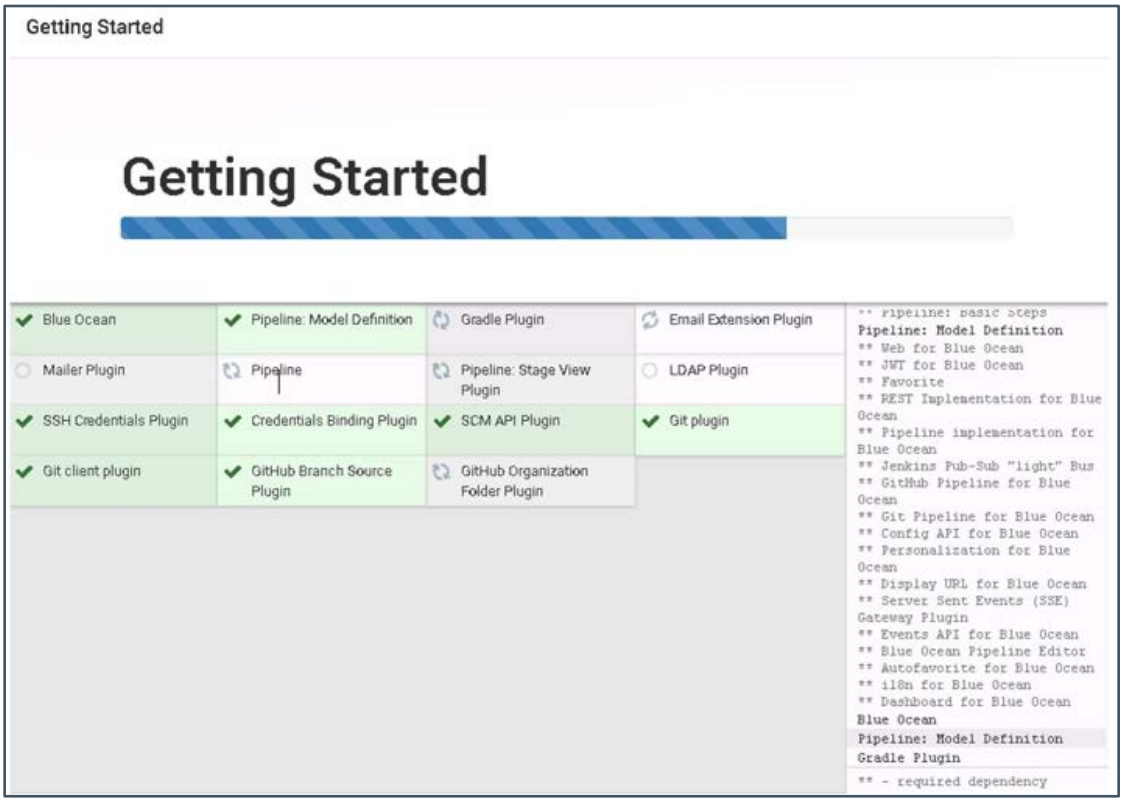

Figure 35. Installing plugins

It is recommended that you create a new administration account, using the dialog shown in Figure 36.

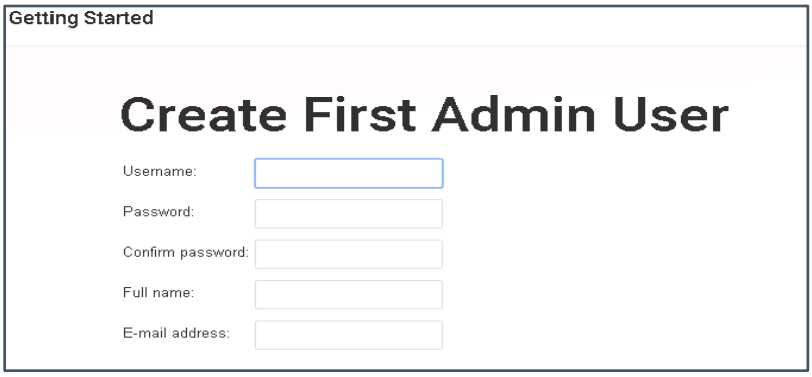

Figure 36. Create Admin user

When the installation and configuration of CloudBees Jenkins Team Edition is complete, the Welcome to Jenkins screen will appear, as show in Figure 37.

Figure 37. Welcome to Jenkins

Instance administration for CloudBees Jenkins Team Edition is documented at https://go.cloudbees.com/docs/cloudbees-documentation/admin-instance/, covering management of upgrades/plugins, securing an instance and common configuration options.

Documentation for administering CloudBees Jenkins Enterprise docs is available at https://go.cloudbees.com/docs/cloudbees-documentation/admin-cje/.

Appendix C: ELK Stack Deployment workflow

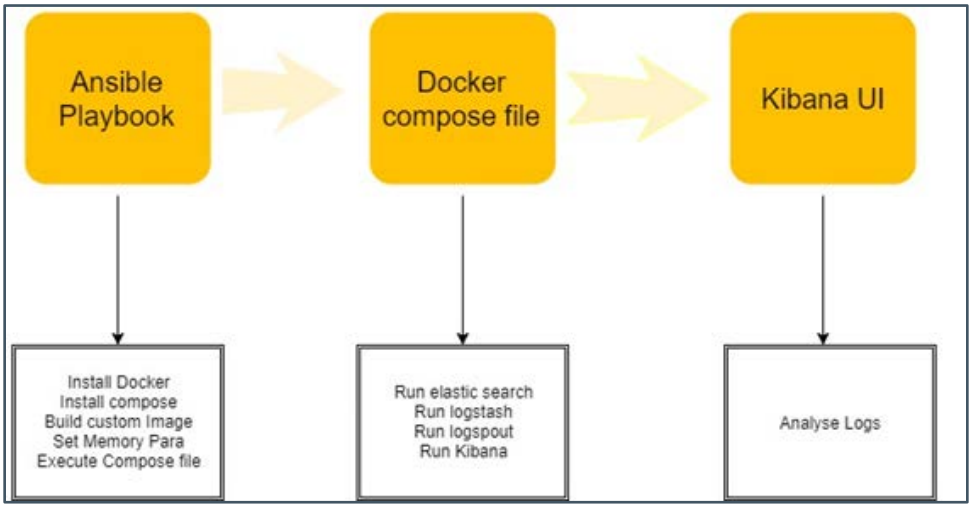

Figure 38. ELK Deployment

Kibana configuration

Kibana is used for data visualization. Kibana queries Elasticsearch periodically and helps a user create monitoring dashboard. The following figures show how to setup the index and discover data. Please note that the default port for Kibana, 5601, is mapped to port 80 on the ELK node. As a result, you can access the Kibana dashboards by accessing the VM for the ELK node on the standard port 80.

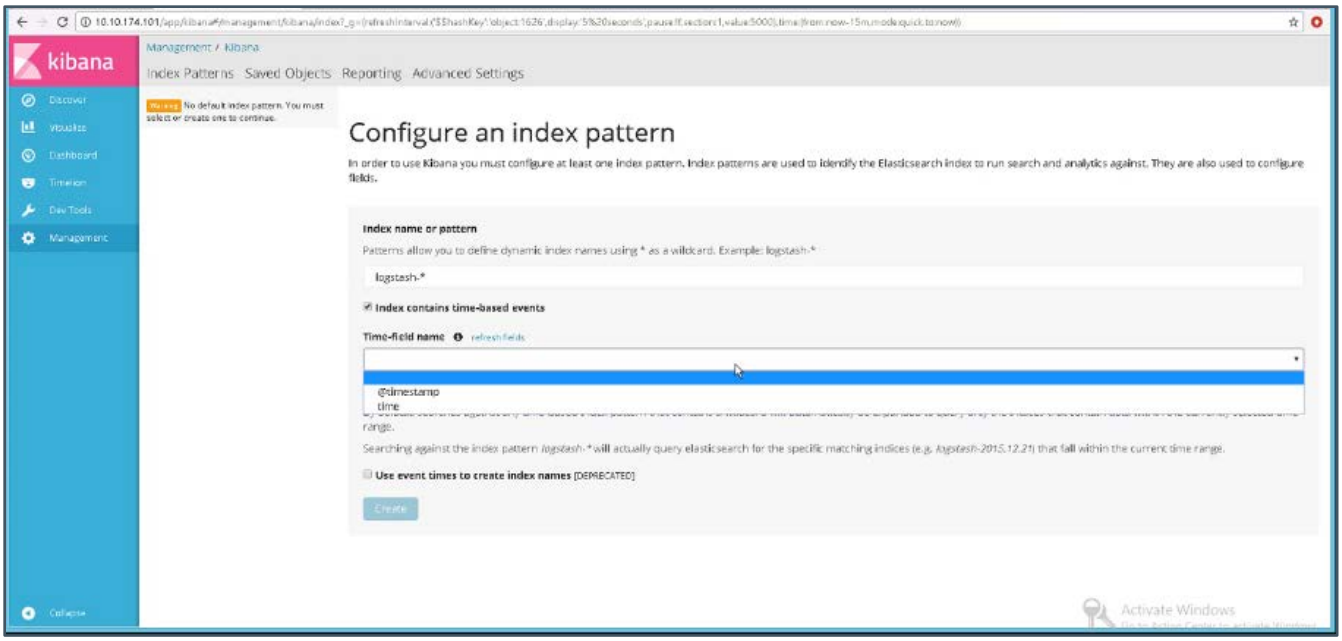

Figure 39. Configure an index pattern in Kibana

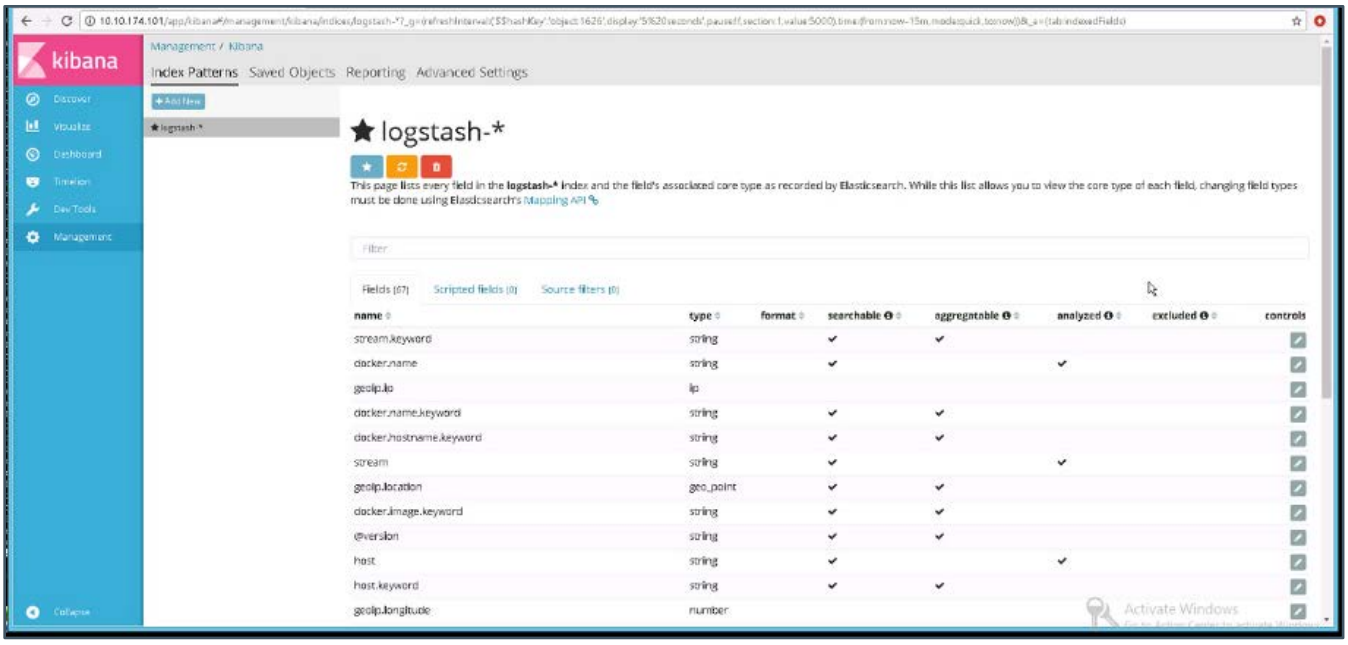

Figure 40. Logstash index, showing the mappings

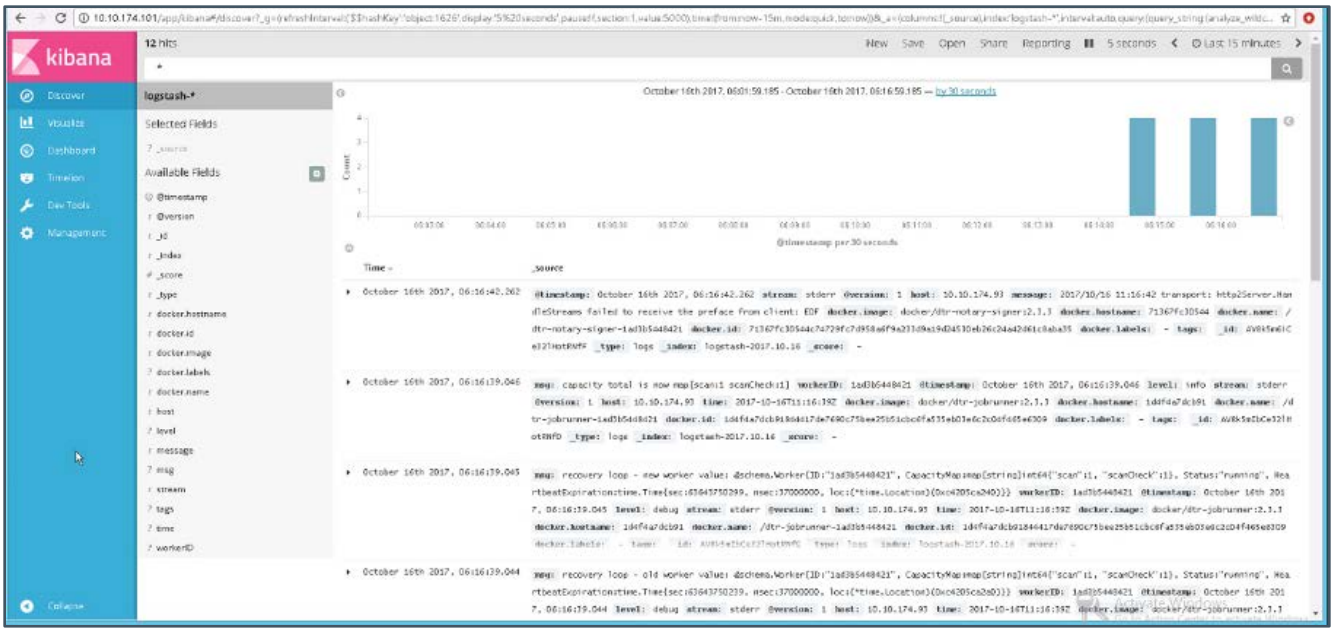

Figure 41. Data discovery in Kibana

For more information on building dashboards, please see the relevant Kibana documentation at https://www.elastic.co/guide/en/kibana/current/dashboard-getting-started.html

The following configuration files are available for the ELK stack:

- `logstash.conf` specifies input plugins, filters and output plugins for the stack.
- `elk-docker-compose.yml` defines ELK container services which will be started on ELK Node.

Appendix D: Play with Docker Customization and Installation

The Play with Docker instance is deployed via the `install_playwithdocker.yml` Ansible playbook. The version of PWD included with this solution has been customized to use a 24-hour session timeout value by default, as we believe this offers the most flexibility for enterprise customers interested in testing application code in a Docker environment.

To modify the PWD session timeout value, change the following variable in the `group_vars/vars` file:

```
pwd_duration: '24h'
```

For example, to deploy a PWD session with a 12-hour timeout value, change this variable as follows:

```
pwd_duration: '12h'
```

To deploy a new PWD session using the modified session timer, re-run the `install_playwithdocker.yml` playbook. Once the session is deployed, launch a web browser on the system where PWD is running and browse to `localhost` as shown in Figure 42:

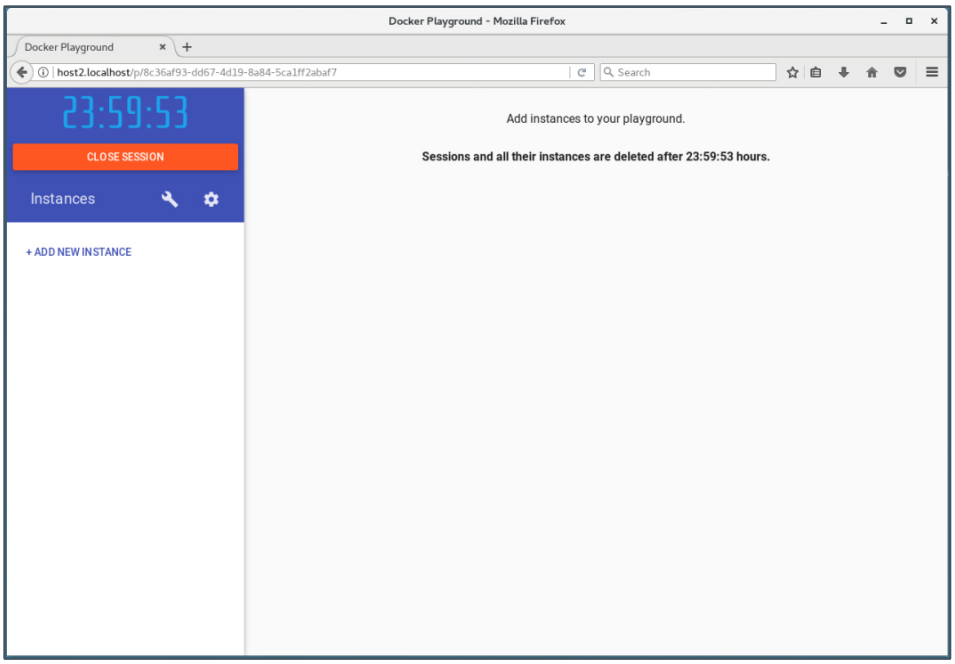

Figure 42. Play with Docker user interface

Selecting the wrench icon opens a “Templates” box allowing users to choose between 3 Docker Swarm Manager nodes and 2 Worker nodes or 5 Manager nodes and no Workers, as shown in Figure 43:

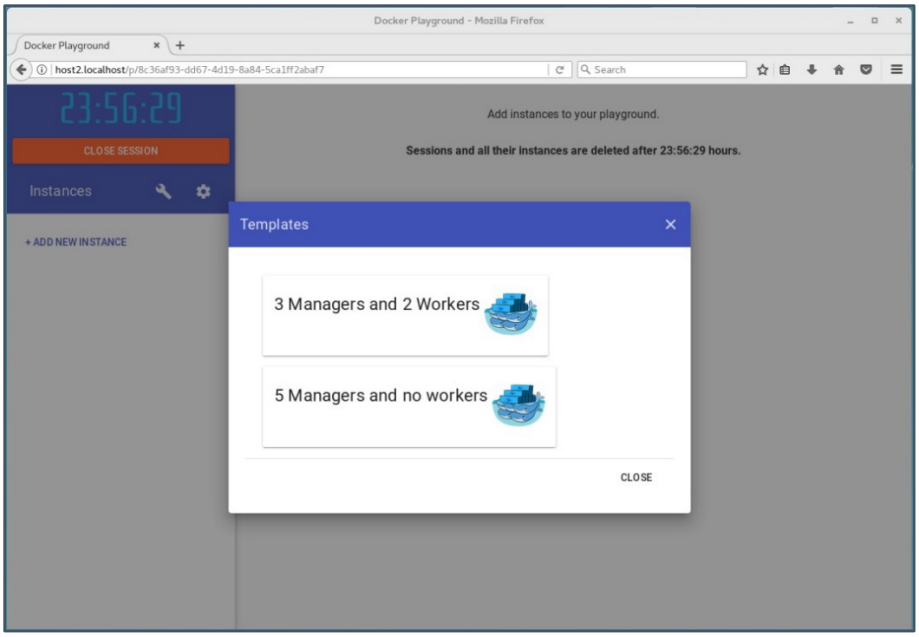

Figure 43. Configure Play with Docker

For more information about using the Play with Docker service, visit the Play with Docker Classroom site at: http://training.play-with-docker.com.

Resources and additional links

HPE Reference Architectures
hpe.com/info/ra

HPE | Docker Alliance page
http://h22168.www2.hpe.com/us/en/partners/docker/

HPE Servers
hpe.com/servers

HPE Storage
hpe.com/storage

HPE Networking
hpe.com/networking

HPE Technology Consulting Services
hpe.com/us/en/services/consulting.html

Sign up for updates

Hewlett Packard Enterprise