Executive Summary

HPE Enterprise Containers as a Service 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 Synergy 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 Enterprise Containers as a Service 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 Enterprise Containers as a Service with Docker EE provides a solution for IT operations, addressing 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 HPE Enterprise Containers as a Service with Docker EE. 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.

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

Assumptions: The present document assumes a minimum understanding in concepts such as virtualization and containerization and also some knowledge around Linux®, Microsoft Windows® and VMware® technologies.

Solution overview

The HPE Enterprise Containers as a Service with Docker EE solution consists of a set of Ansible playbooks that run on top of a VMware virtualization platform on HPE Synergy hardware. The solution allows you to configure a flexible OS environment (with both RHEL and Windows workers) providing built-in high availability (HA), container monitoring and security, and backup and restore functionality. This solution assumes that you have already set up your HPE Synergy hardware, that you have installed your VMware virtualization platform and have configured HPE 3Par for storage.

Solution configuration

By default, the Ansible playbooks are configured to set up a 3 node environment as this is the minimal starter configuration as recommended by HPE and Docker for production. However, the playbooks can be configured to fit your environment and your high availability (HA) needs and the solution has been tested on a 6 node HPE Synergy environment, with 2 nodes in each frame.

Two separate configurations are available out of the box, with one restricted to a Linux-only deployment while the other supports a hybrid deployment including Windows workers as well as Linux ones. The Docker and non-Docker modules are distributed over the physical nodes via virtual machines (VMs), depending on the size of your environment, as follows:

- 3 Docker Universal Control Plane (UCP) VM nodes for HA and cluster management
- 3 Docker Trusted Registry (DTR) VM nodes for HA of the container registry

The Docker UCP and DTR nodes are spread across 3 physical nodes, with one on each physical node. An odd number of manager nodes is recommended to avoid split-brain issues. It is possible to restrict the deployment to 1 UCP and 1 DTR, or to expand to more than 3, but the recommended minimum for an enterprise production deployment is 3 UCPs and 3 DTRs.

- 3 Docker Swarm Linux worker VM nodes for container workloads
- 3 Docker Swarm Windows worker VM nodes for container workloads (optional)

The Docker worker nodes will be co-located with the UCP and DTR nodes in a 3 physical node deployment, whereas in a 6 physical node set-up, the worker nodes will typically be separated onto the extra nodes. It is possible to specify that more than one worker node will be deployed per physical node but this decision will depend on the requirements of your applications.

- 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

Three load balancers are provided to increase availability of the UCP, DTR and worker nodes and these are typically distributed evenly across 3 physical nodes.

- 1 Logging server VM for central logging
- 1 NFS server VM for storage of Docker DTR images

With the addition of the NFS and logging VMs, a total of 14 VMs are created for the default Linux-only deployment. The hybrid deployment will add 3 Windows worker nodes to this figure. In addition to these VMs, the playbooks also set up the Docker persistent storage plug-in from VMware. The vSphere Docker volume plug-in facilitates the storage of data in a shared datastore that can be accessed from any machine in the cluster.

High availability

Uptime is paramount for any users implementing Docker containers in business critical environments. HPE Enterprise 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. The Ansible playbooks can be modified to fit your environment and your high availability (HA) needs.

Load Balancers

HPE Enterprise 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. Since these 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/.

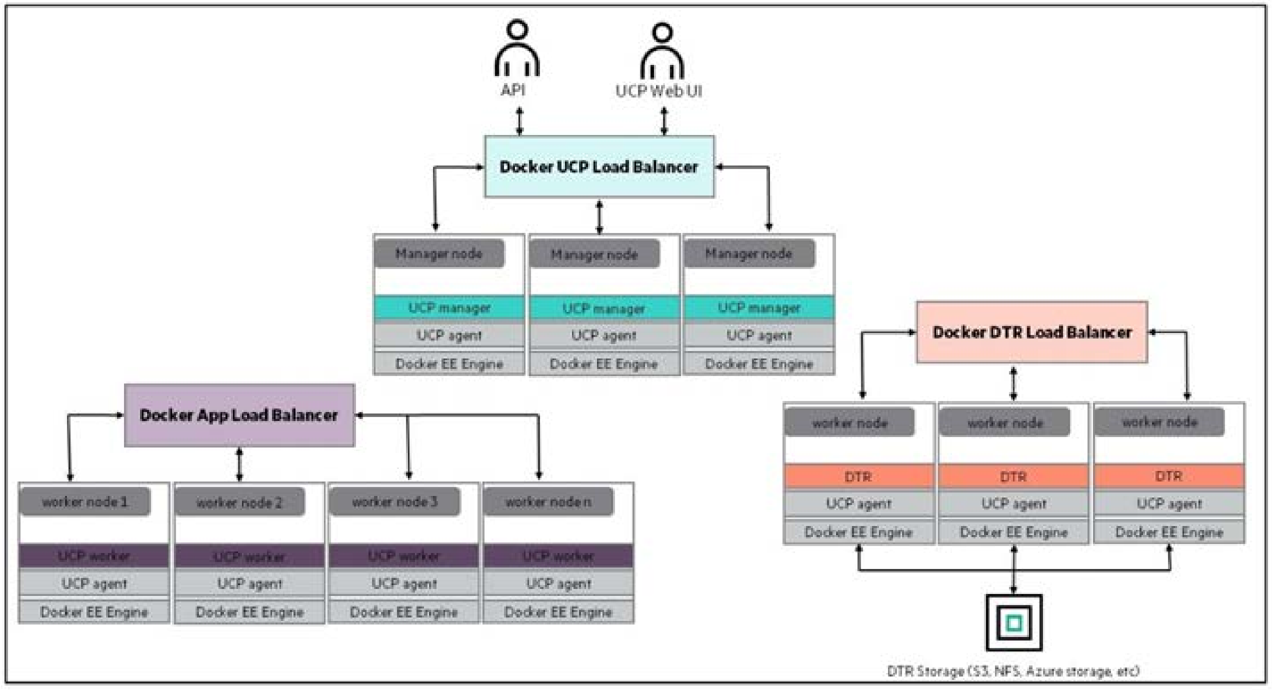


Figure 1. Load balancer architecture

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 workloads; 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 Enterprise Containers with Docker EE: Ops 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	node01	node02	node03
ucp1	4		
ucp2		4	

ucp3			4
dtr1	2		
dtr2		2	
dtr3			2
worker1	4		
worker2		4	
worker3			4
win-worker1	4		
win-worker2		4	
win-worker3			4
ucb_lb	2		
dtr_lb		2	
worker_lb			2
nfs			2
logger		2	
Total vCPU per node	12	14	14

Note

In the case of one ESX host failure, two nodes are enough to accommodate the amount of vCPU required

Table 2. Memory allocation

RAM (GB)	node01	node02	node03
ucp1	8		
ucp2		8	
ucp3			8
dtr1	16		
dtr2		16	
dtr3			16
worker1	64		
worker2		64	
worker3			64
win-worker1	64		
win-worker2		64	
win-worker3			64
ucb_lb	4		
dtr_lb		4	
worker_lb			4
nfs			4
logger		4	
Total RAM required (per node)	92	96	96
Total RAM required		284	

Available RAM	384	384	384

Note
In the case of one ESX host failure, the two surviving hosts can accommodate the amount of RAM required for all VMs.

Disaster Recovery

Recovery Time Objective (RTO) refers to the time that it takes to recover your data and applications while Recovery Point Objective (RPO) refers to the point in time you can recover to in the event of a disaster. In essence, RPO tells you how often you will need to make new backups.

In order to protect your installation from disasters, you need to take regular backups and transfer the backups to a safe location. This solution provides a range of convenience scripts and Ansible playbooks to help automate the backup of UCP, DTR, your swarm and your Docker volumes. See the section Backup and restore for best practices, procedures and utilities for implementing disaster recovery.

Security

The Docker Reference architecture for Securing Docker EE and Security Best Practices is available at https://success.docker.com/article/Docker_Reference_Architecture-_Securing_Docker_EE_and_Security_Best_Practices

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.

Solution components

Hardware

Table 3 lists the hardware components that are utilized in this Reference Configuration

Table 3. Hardware

Component	Purpose
HPE Synergy 12000 Frame	Rack enclosure for compute, storage, and network hardware
HPE Synergy 480 Gen10 Compute Modules	Hosts for running ESX servers that support UCP, DTR, worker and other nodes in the solution
HPE 3PAR StoreServ	Provides the storage for the virtual machines and the Docker backups
HPE StoreOnce	High performance backup system

HPE Synergy

HPE Synergy, the first platform built from the ground up for composable infrastructure, empowers IT to create and deliver new value instantly and continuously. This single infrastructure reduces operational complexity for traditional workloads and increases operational velocity for the new breed of applications and services. Through a single interface, HPE Synergy composes compute, storage and fabric pools into any configuration for any application. It also enables a broad range of applications from bare metal to virtual machines to containers, and operational models like hybrid cloud and DevOps. HPE Synergy enables IT to rapidly react to new business demands.

HPE Synergy Frames contain a management appliance called the HPE Synergy Composer which hosts HPE OneView. HPE Synergy Composer manages the composable infrastructure and delivers:

- Fluid pools of resources, where a single infrastructure of compute, storage and fabric boots up ready for workloads and demonstrates self-assimilating capacity.

- Software-defined intelligence, with a single interface that precisely composes logical infrastructures at near-instant speeds; and demonstrates template-driven, frictionless operations.
- Unified API access, which enables simple line-of-code programming of every infrastructure element; easily automates IT operational processes; and effortlessly automates applications through infrastructure deployment.

Server requirements
The minimum platform requirement for this configuration is a 3 node HPE Synergy 480 Gen10 deployment with 1 node in each Synergy frame and

- 384 GB DDR4-2133 RAM
- 2 Intel Xeon CPU Gold 6130 2.10GHz x 16 core
- Single ESXi cluster with control plane and Docker workers spread out on all 3 nodes

The solution has also been tested on a 6 node HPE Synergy environment, with 2 nodes in each frame. In this setup, the extra 3 nodes are dedicated to Docker worker nodes. The 6 node deployment is depicted graphically in Figure 2.

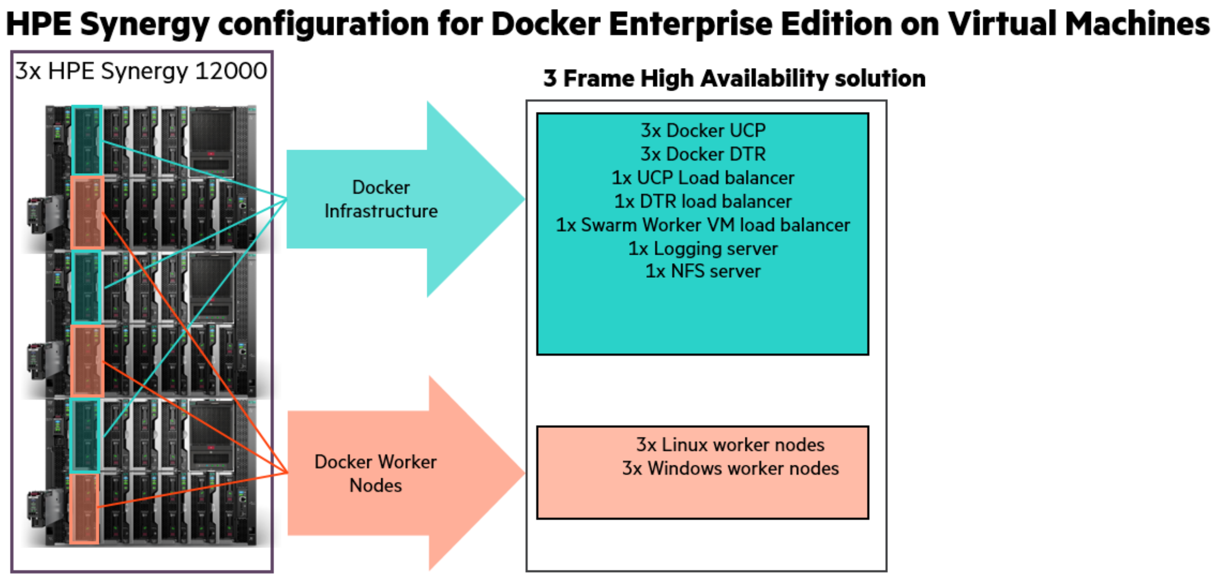


Figure 2. HPE Synergy Configuration

Storage requirements
A HPE 3PAR store is required for ESXi datastore. This solution makes use of a HPE 3PAR StoreServ 8200 populated with:

- 8x480GB SSD for the vSphere cluster datastore
- 8x1.8TB HDD for Backup Datastore

You should create a large virtual volume on the 3PAR StoreServ to host the virtual machines and another large virtual volume for Docker backups. Create datastores on your vSphere cluster using these virtual volumes. If desired, you can create separate StoreServ virtual volumes and attach them to all vSphere cluster hosts for backing up Docker persistent volumes. It is recommended that you configure the volumes that are used for virtual machine deployments on the SSD. Storage for backups can be configured on the HDDs.

Software
The software components used in this Reference Configuration are listed in Table 4 and Table 5.

Table 4. Third-party software

Component	Version
Ansible	2.4.2
Docker EE	17.06 (tested with UCP 2.2.7 and DTR 2.4.3)
Red Hat Enterprise Linux	7.4
Microsoft Windows	Server 2016
VMWare	ESXi 6.5.0 and vCenter 6.5.0

Table 5. HPE Software

Component	Version
HPE Recovery Manager Central	5.0.1

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.

When deploying Windows nodes in a hybrid deployment, the Ansible playbooks make use of the Python `pywinrm` module which carries out actions via the Windows remote manager.

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

About Docker Enterprise Edition

Docker Enterprise Edition (EE) is a 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

Application software

A number of different logging and monitoring solutions are supported by this solution:

- Splunk®
- Sysdig®

- Prometheus and Grafana

The application software components used in this Reference Configuration are listed in Table 6.

Table 6. Application software

Component	Version
Splunk	7.0.3
Sysdig	latest
Prometheus	v1.7.1
Grafana	4.4.3

Monitoring with Splunk and Sysdig

The solution can be configured to use of either Splunk or Sysdig or to enable both simultaneously. While there is some overlap in the functionality provided by these tools, they are ultimately complimentary in what they offer. Splunk aggregates logging and tracing for a wide variety of sources and provides a clean, high-level dashboard for all your enterprise systems. Sysdig, on the other hand, has been engineered from the ground up to focus on containerized environments and includes both monitoring and security features, with built-in understanding of the different workloads running on your cloud.

The load among the three hosts in a hybrid deployment will be shared as per Figure 3:

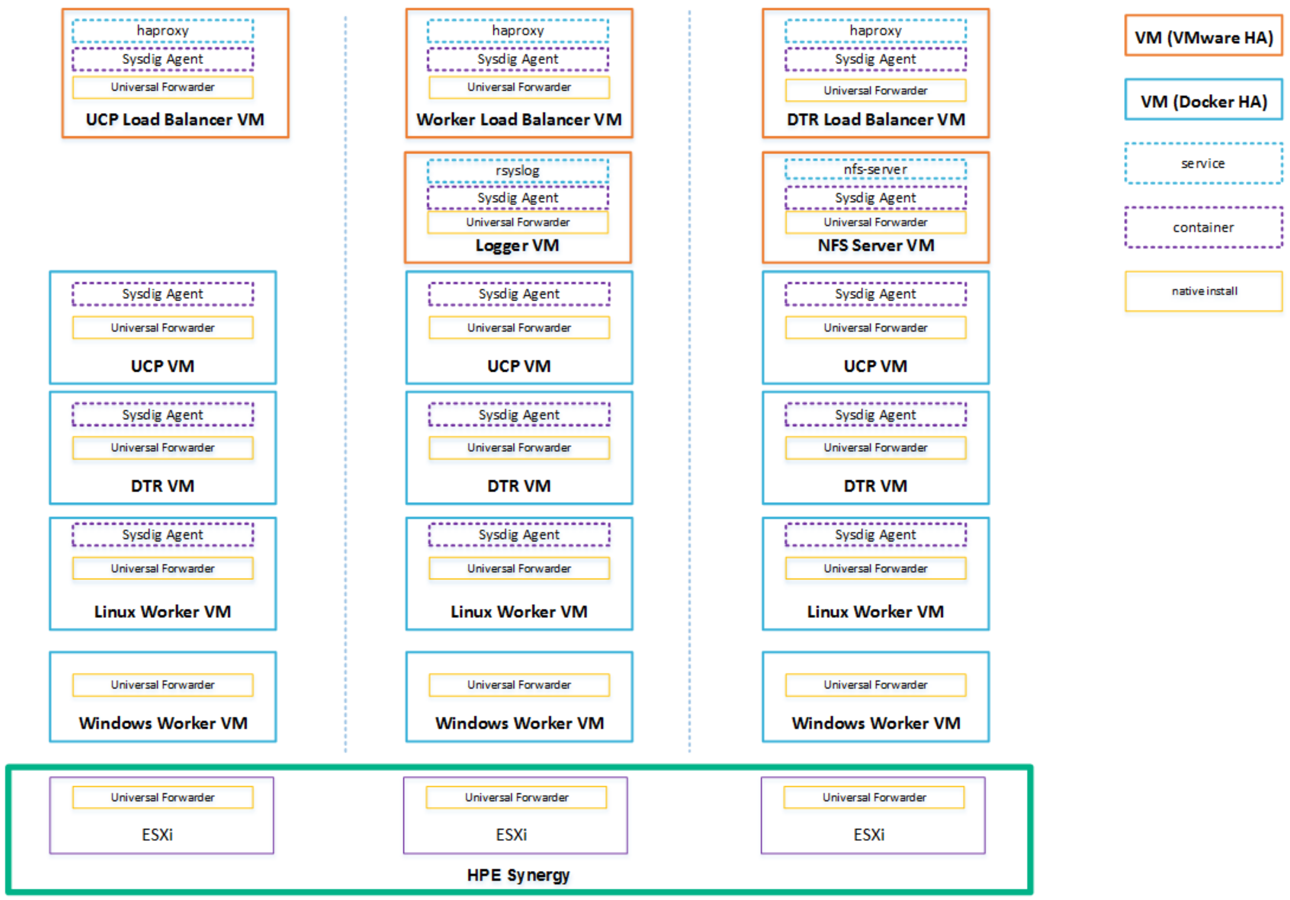


Figure 3**.** Solution architecture: Hybrid Linux and Windows workers with Splunk and Sysdig

Monitoring with Splunk

Splunk Enterprise allows you to collect and index any data from any source, and to monitor systems and infrastructure in real time to preempt issues before they happen. It allows you to analyze your data to understand trends, patterns of activity and behavior, giving you valuable intelligence across your entire organization.

This solution allows you to integrate your Container as a Service deployment with an existing Splunk Enterprise installation or to deploy a stand-alone Slunk demo environment as a Docker stack in your cloud. In both instances, Universal Forwarders are used to collect data from your applications running on your Linux and Windows worker nodes in your cloud, as well as log data from the Docker platform itself and from the infrastructure VMs and servers.

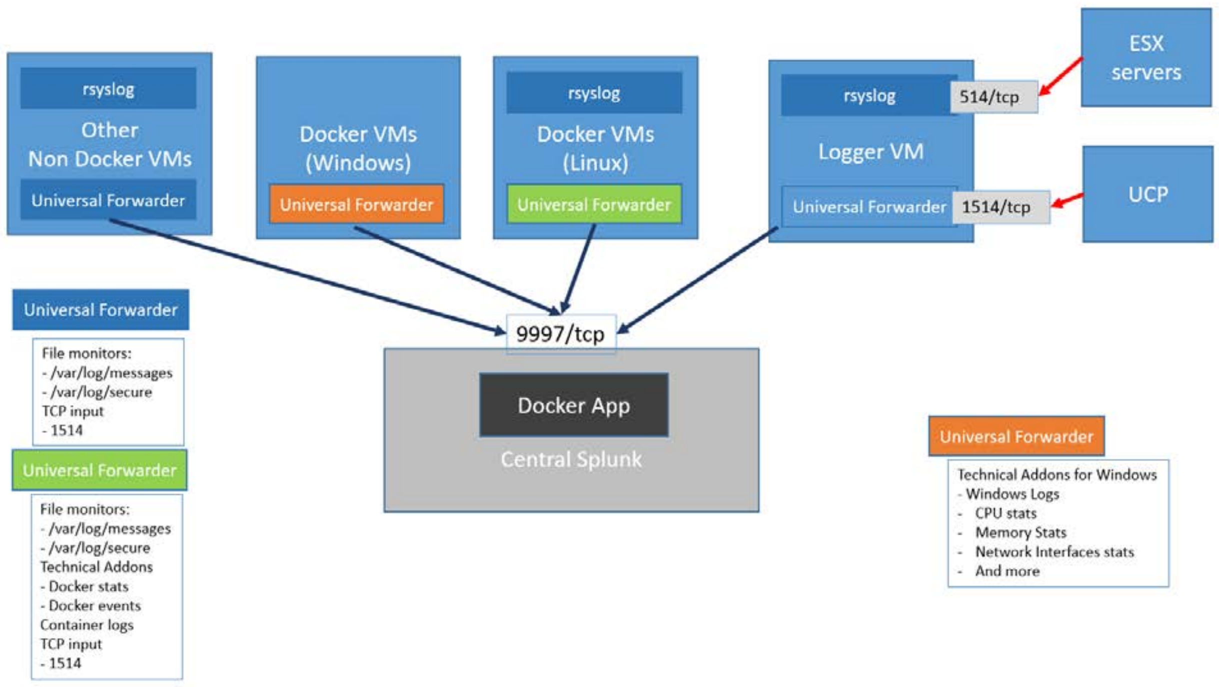


Figure 4**.** Splunk architecture

All the Universal Forwarders run natively on the operating system to allow greater flexibility in terms of configuration options. Each forwarder sends the data it collects to one or more indexers in the central Splunk.

Linux worker nodes: The Universal Forwarders on the Linux worker nodes collect log and metrics data. The log data includes:

- `/var/log/messages` from the docker host (including the daemon engine logs)
- `/var/log/secure` from the docker hosts
- container logs via a Splunk technical add-on

The metrics data is collected via a technical add-on and includes:

- `docker stats`
- `docker top`
- `docker events`
- `docker service stats`

Windows worker nodes: The Universal Forwarders running on the Windows worker nodes collect the following data:

- Windows logs
- CPU stats
- Memory stats
- Network Interface stats
- and more

For more information on configuring standalone Splunk for Linux and Windows worker nodes, see the section on Splunk prerequisites.

UCP and ESXi: UCP operational logs and ESXi logs are forwarded to the logger VM via TCP ports 1514 and 514 respectively. Port 1514 is assigned a special `sourcetype` of `ucp` which is then used by the Splunk Docker APP to interpret UCP logs. The Universal Forwarder runs the rsyslog daemon which will record the log messages coming from the ESX machines into the `/var/log/messages` file on the VM.

Non-Docker VMs: Other VMs, for example, NFS, use a Splunk `monitor` to collect and forward data from the following files:

- /var/log/messages
- /var/log/secure (Red Hat)

Note
You can configure the list of files monitored by the Universal Forwarder.

Other syslog senders can be configured to send their data to the logger VM or directly to central Splunk.

Monitoring with Sysdig

Sysdig's approach to Docker monitoring uses transparent instrumentation to see inside containers from the outside, with no need for agents in each container. Metrics from Docker containers, and from your applications running inside them, are aggregated in real-time across each service to provide meaningful monitoring dashboards and alerts for your application.

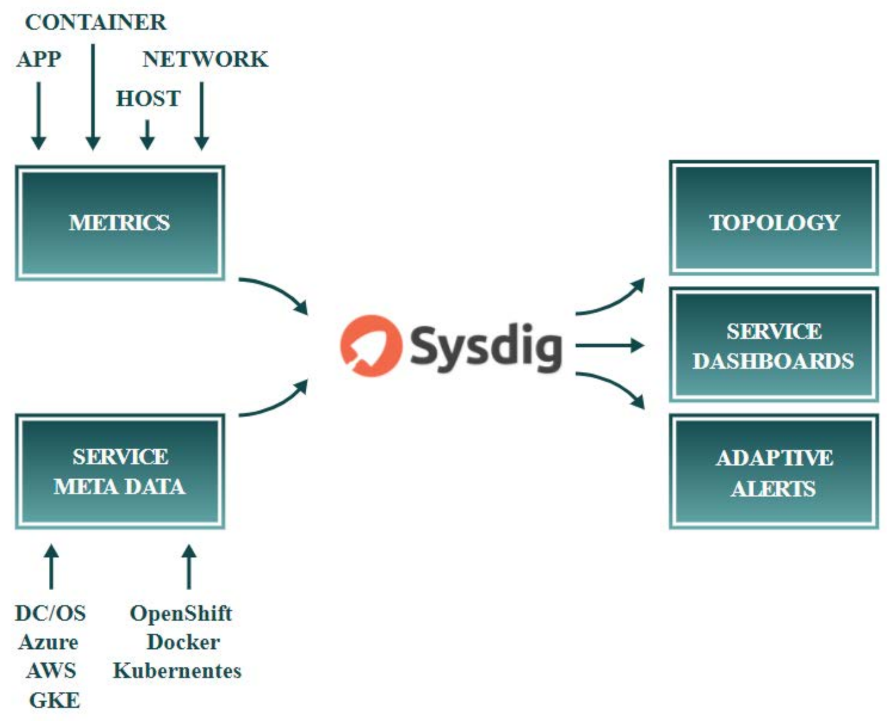


Figure 5**.** Sysdig architecture

Sysdig Monitor allows you to analyze response times, application performance metrics, container and server utilization metrics, and network metrics. You can build dashboards across applications, micro-services, container and networks, and explore metadata from Docker, Kubernetes, Mesos and AWS.

Sysdig Secure provides security at the orchestrator as well as the container level. You create service-aware policies that allow you to take actions (like killing a container) or send alerts (to Slack, Splunk, etc) whenever a policy violation occurs. All commands are audited to help you identify anomalous actions, along with taking snapshots of all activities pre-and-post a policy violation.

The implementation in this solution uses the Software as a Service (SaaS) version of Sysdig. The playbooks deploy a Sysdig Agent software on each UCP, DTR and Linux worker node, as well as the NFS, logger and load balancer VMs and captured data is relayed back to your Sysdig SaaS Cloud portal.

Note
The Sysdig functionality is not turned on by default in this solution - see the section on Configuration for more information on how to enable Sysdig.

Monitoring with Prometheus and Grafana

The solution can be configured to enable the use of Prometheus and Grafana for monitoring. The load among the three hosts will be shared as per Figure 6:

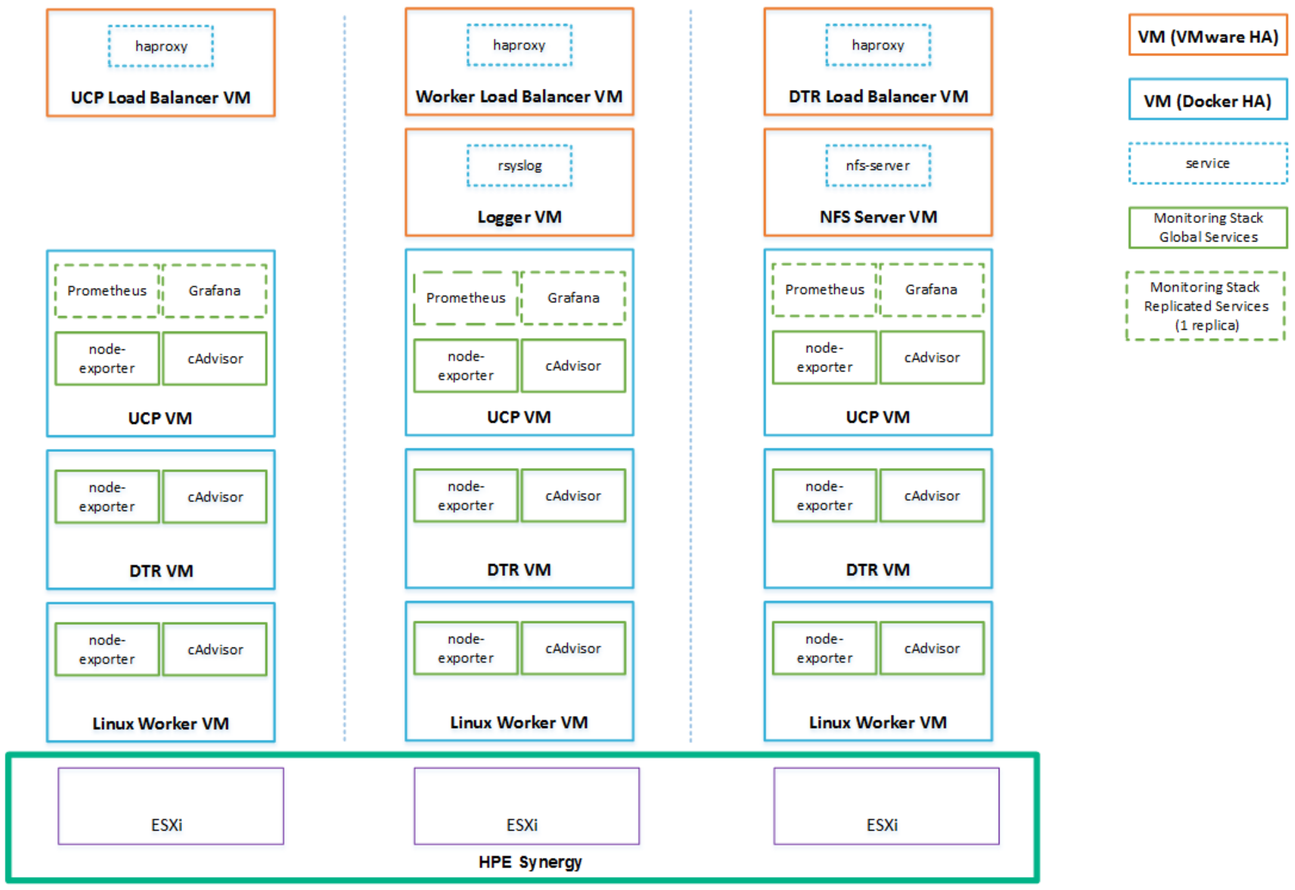


Figure 6**.** Solution architecture: Linux workers with Prometheus and Grafana

The Prometheus and Grafana services are declared in a Docker stack as replicated services with one replica each, so if they fail, Docker EE will ensure that they are restarted on one of the UCP VMs. cAdvisor and node-exporter are declared in the same stack as global services, so Docker EE will ensure that there is always one copy of each running on every machine in the cluster.

Preparing the environment

This section describes in detail how to prepare the environment that was outlined in the architecture section. The following high level steps are required:

- Verify prerequisites
- Enable vSphere High Availability (HA)
- Install vSphere Docker Volume Service driver on all ESXi hosts
- Create the Ansible node
- Create the Red Hat Linux Template and configure the `yum` repositories
- Create the Windows Template (optional)
- Finalize the template

Verify prerequisites

Before you start deployment, you must assemble the information required to assign values for each and every variable used by the playbooks. The variables are fully documented in the section Configuring the solution components. A brief overview of the information required is presented in Table 7.

Table 7. Summary of information required

Component	Details
Virtual Infrastructure	The FQDN of your vCenter server and the name of the Datacenter. You will also need administrator credentials in order to create templates and spin up virtual machines.
L3 Network requirements	You will need one IP address for each and every VM configured in the Ansible inventory (see the section Configuring the solution components). The recommended minimal deployment (Linux-only) configures 14 virtual machines so you would need to allocate 14 IP addresses to use this example inventory. If you have a hybrid environment with Windows workers, you will need to increase the allocation. 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 to know 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 deployed solution uses certificates that are time-sensitive. You will need to specify 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 (HA)

You must enable vSphere High Availability (HA) to support virtual machine failover during a 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.

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. Setting this up 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 and then 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 at http://vmware.github.io/vsphere-storage-for-docker/documentation/install.html

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 the Ansible node

The Ansible node will act as the driver to automate the provisioning of the environment and it is essential that it is properly installed.

Create a Virtual Machine and install your preferred OS (in this example, and for the sake of simplicity, RHEL7 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.

Log in to 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
```

Configure the following yum repositories, `rhel-7-server-rpms` and `rhel-7-server-extras-rpms` as explained in Configure the yum repositories. The "extras" repo can be enabled as follows:

```
# subscription-manager repos --enable=rhel-7-server-extras-rpms
```

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
```

Install Ansible 2.4.2

```
# yum install ansible
```

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
# pip install ”pywinrm>=0.2.2”
```

Configure the yum repositories

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 is on the Red Hat DVD but the second is not. There are two options, with both options requiring a Red Hat Network account. Logon in your VM template using SSH with the credentials you configured for the root account and implement one of the two options below:

Option 1: 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. Use the `subscription-manager register` command as follows.

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

If you are behind a proxy, you must configure this before running the above command to register.

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

Verify that you don't have the issue described here: https://access.redhat.com/solutions/3317671 by entering the following command.

```
# yum repolist
```

If you have the issue, fix it with the following command

```
# subscription-manager repos --disable=rhel-7-server-rt-beta-rpms
```

The playbooks will later automatically enable the `extras` repository on the VMs that need it.

Option 2: 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 as follows.

```
[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
```

To see how you can create a local mirror of the Red Hat repositories and how to share them, check the Red Hat documentation at https://access.redhat.com/solutions/23016 and at https://access.redhat.com/solutions/7227.

Create the Red Hat Linux template

To create the Red Hat Linux VM template that you will use as the base for all your nodes, you first create a Virtual Machine with the OS installed and then convert the Virtual Machine to a VM Template. The VM Template is created as lean as possible, with any additional software installs and/or system configuration performed subsequently using Ansible.

As the creation of the template is a one-off task, this procedure has not been automated. The steps required to manually create a VM template are outlined below.

Log in to vCenter and create a new Virtual Machine with the following characteristics:

- Guest OS Family: Linux, Guest OS Version: Red Hat Enterprise Linux (64-bit)
- Hard Disk size: 50GB, (Thin provisioning)
- A single network controller connected to the network or VLAN of your choice. All VMs will connect to this same network.
- Optionally you can remove the floppy drive

Install Red Hat Enterprise 7:

1. Select a language which is selected by Docker

2. For the software selection, choose **Infrastructure Server** as the base environment and add the **Guest Agents** from the lists of add-ons available for this environment. The Infrastructure Server environment is selected here versus the Minimal Install because Customization of Linux guest operating systems requires that Perl is installed in the Linux guest operating system.
3. Configure the network settings so that you can later access the VM using SSH. Specify an IP address for the network interface, a default gateway, DNS settings and possibly any HTTP/HTTS proxies that apply in your environment.
4. Specify a password for the root account and optionally created an admin user.
5. Wait for the installation to finish and for the VM to reboot.

Finalize the template
Log in to the `root` account on the Ansible box and copy the SSH public key to the VM Template. This will allow your Ansible node to SSH to all the Virtual Machines created from the VM Template without the need for a password.

```
ssh-copy-id root@<IP of your VM_Template>
```

Perform the following steps on the VM Template to finalize its creation:

1. Clean up the template by running the following commands from the **Virtual Machine Console:**

```
# rm /etc/ssh/ssh_host_*
# nmcli con del ens192
# logrotate -f /etc/logrotate.conf
# rm /var/log/*-201?*
# history -c
```

2. Shutdown the VM

```
# shutdown -h now
```

3. Turn the VM into a template by right-clicking on your VM and selecting `Template -> Convert to Template`. This will create a new template visible under VM Templates in Folders, ready for future use.

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

Create the Windows Template (optional)
To create the Windows VM Template that you will use as the base for all your Windows worker nodes, you will first create a Virtual Machine with the OS installed and then convert the Virtual Machine to a VM Template. The VM Template is created as lean as possible, with any additional software installs and/or system configuration performed subsequently using Ansible.

As the creation of the template is a one-off task, this procedure has not been automated. The steps to create a VM template manually are outlined below.

Log in to vCenter and create a new Virtual Machine with the following characteristics:

- Guest OS Family: Windows, Guest OS Version: Microsoft Windows Server 2016 (64-bit)
- Hard Disk size: 100GB (Thin provisioning), 1 vCPU and 4 GB of RAM. Both vCPU and memory can be altered later after you deploy from this template.
- A single network controller connected to the network or VLAN of your choice. All VMs will connect to this same network.

- Change the network type to VMXNET3, and attach the Windows 2016 ISO image from a datastore ensuring you connect the CD/DVD drive on boot.
- Click on the VM Options tab, and in the Boot Options section, select `Force BIOS setup(*)` to ensure that the machine enters the BIOS setup screen on next boot of this VM. This will allow you to adjust the boot order, placing the virtual CDROM in front of your hard drive.
- Optionally you can remove the floppy drive.

Install Windows 2016:

- Power on the selected VM and then `Open Console`.
- Once connected to the console, you will be placed in the BIOS setup screen. Select the `Boot` tab, click on CD-ROM Drive and move up the CDROM drive above the hard drive. This allows your Windows 2016 ISO image to be loaded first on boot. F10 Save and exit is next step.
- Enter your choices for Language, Time/Currency Format, Keyboard and then Install Now.
- Select the OS you want to install, and then select Custom: Install Windows Only.
- Select drive 0, the 100 GB drive you specified earlier, as the location for installing windows.
- Add a password for the Administrator user.
- Install VMware Tools and reboot.

Once the VM has re-booted:

- Add a temporary network IP address.
- Use sconfig utility from (MS-DOS) command line to install windows updates and enable remote desktop.
- Perform any other customizations you require at this point.

Prior to converting the VM to Template, run Sysprep: `C:\Windows\System32\Sysprep\Sysprep.exe`

- Ensure 'System Out-of-Box Experience (OOBE)' is selected
- Select the 'Generalize' option
- Select 'Shutdown' from the Shutdown Options.

Shutdown VM, and untick `Connect CD/DVD` so that the Windows 2016 ISO is no longer mounted. Boot the Windows VM one final time and enter regional settings applicable to your location and keyboard mapping, then Shutdown VM.

Note
The `vmware_guest` module used by the playbooks will generate a new SID.

Turn the VM into a template by right-clicking on your VM and selecting `Template -> Convert to Template`. This will create a new template visible under VM Templates in Folders, ready for future use.

Configuring the solution components

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`. The variables can be defined in any order but for the sake of clarity they have been divided into sections.

Ansible configuration

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

```
# git clone <repository>
```

Change to the directory which you just cloned:

```
# cd ~/Docker-Synergy
```

Change to the `ops` directory:

```
# cd ops
```

Note
All subsequent file names are relative to the `ops` directory. For example `vm_hosts` is located in `~/Docker-Synergy/ops` and `group_vars/vars` corresponds to `~/Docker-Synergy/ops/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 modify a number of files including:

- `site.yml`, the main entry point for the playbooks.
- `vm_hosts`, the inventory file.

You also need to create and populate a number of files:

- `group_vars/vars`, the group variables file.
- `group_vars/vault`, containing sensitive information that needs to be protected.
- `group_vars/backup`, containing backup-related variables.

For the latter group, a set of sample files has been provided to help you get started:

- `group_vars/vars.sample`, a sample group variables file.
- `group_vars/vault.sample`, a sample vault file.
- `group_vars/backup.sample`, a sample backup configuration file.

The file `group_vars/win_worker.yml` supports advanced configuration of Windows remote management and in general should not require modification.

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

Editing the inventory

The inventory is the file named `vm_hosts` in the `~Docker-Synergy/ops` 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. Other fields (hostnames, specifications, IP addresses…) are edited to match your setup. 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. Typically you should have either 3 or 5 nodes under this group.
- `[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. Typically you should have either 3 or 5 nodes under this group.
- `[worker]`: A group containing all the Linux worker nodes.
- `[win_worker]`: A group containing all the Windows worker nodes.
- `[ucp_lb]`: A group containing one single node which will be the load balancer for the UCP nodes. 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 nodes. 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 individually 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.
- `[local]`: A group containing the local Ansible host. It contains an entry that should not be modified.

There are also a few 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, with the exception of 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.
- [win_worker:vars]: A set of variables defined for all nodes in the `[win_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.

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 Linux 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'
worker03 ip_addr='10.0.0.12/16' esxi_host='esxi1.domain.local' cpus='16' ram'32768'

[worker:vars]
cpus='4' ram='16384' disk2_size='200'
```

In the example above, the `worker03` node would have 4 times more CPU and double the RAM compared to the rest of the worker nodes.

The different variables you can use are as described in Table 8 below. They are all mandatory unless otherwise specified.

Table 8. 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.

VMware configuration

All VMware-related variables are mandatory and are described in Table 9.

Table 9. 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 RHEL VM Template to be use. 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`'...]. This or these datastore(s) 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 0.20 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.
vm_portgroup	Used by the playbook `create_vms.yml`, this variable is used to specify the portgroup connected to the network that connects all the VMs. There is currently only one network. It is recommended that the template which is used as the base for all deployed VMs specifies a network adapter but it is not required. If a network adapter is specified, you should not attach this adapter to a standard switch if the portgroup designated by `vm_portgroup` is connected to a distributed vSwitch. In addition, you should make sure that the adapter specifies `Connect At Power On`.

Networking configuration

All network-related variables are mandatory and are described in Table 10.

Table 10. 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. ['10.10.173.1','10.10.173.2'…]
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'…]

Environment configuration

All Environment-related variables should be here. All of them are described in Table 11 below.

Table 11. Environment variables

Variable	Description
env	Dictionary containing all environment variables. It contains three entries described below. Please leave empty the proxy related settings if not required: `http_proxy`: HTTP proxy URL, such as `'http://15.184.4.2:8080'`. This variable defines the HTTP proxy url if your environment is behind a proxy. `https_proxy`: HTTPS proxy URL, such as `'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, such as `'localhost,127.0.0.1,.cloudra.local,10.10.174.'`

Docker configuration

All Docker-related variables are mandatory and are described in Table 12.

Table 12. Docker variables

Variable	File	Description
docker_ee_url	**group_vars/vault**	Note: This is a private link to your Docker EE subscription. The value for `docker_ee_url` is the URL documented at the following address: https://docs.docker.com/engine/installation/linux/docker-ee/rhel/.
docker_ee_version	group_vars/vars	If this variable is omitted, `install_docker.yml` will install the latest stable version of docker-ee available in the repo specified with `docker_ee_url` If you want to install a specific version of Docker EE, enter the full specification of the packages, for example: docker_ee_version: 'docker-ee-17.06.2.ee.6-3.el7.rhel.x86_64'
rhel_version	group_vars/vars	For the Docker installation, this sets the version of your RHEL OS, such as `7.4`. The playbooks were tested with RHEL 7.4.
dtr_version	group_vars/vars	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 2.4.3.
ucp_version	group_vars/vars	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 2.2.7.
images_folder	group_vars/vars	Directory in the NFS server that will be mounted in the DTR nodes and that will host your Docker images.
license_file	group_vars/vars	Full path to your Docker EE license file on your Ansible host. The license file is available from the Docker Store
ucp_username	group_vars/vars	Username of the administrator user for UCP and DTR, typically `admin`.
ucp_password	**group_vars/vault**	The password for the `ucp_username` account.

To see how to use customer-supplied certificates with UCP and DTR, see Appendix B.

Windows configuration

Table 13. Windows variables

Variable	File	Description
enable_windows	group_vars/vars	If `true`, the creation of Windows 2016 worker nodes will be actioned. The default value is `false`.
win_vm_template	group_vars/vars	Name of the Windows 2016 VM Template to use. Note that this is the name from a vCenter perspective, not the hostname.
win_username	group_vars/vars	Windows user name. The default is `Administrator`
win_password	**group_vars/vault**	The password for the Windows account.
windows_vdvs_ps	group_vars/vars	Variable used to download the PowerShell script that is used to install vDVS for Windows. For example, `https://raw.githubusercontent.com/vmware/vsphere-storage-for-docker/master/install-vdvs.ps1`
windows_vdvs_path	group_vars/vars	Variable used to download vSphere Docker Volume Service software. This variable is combined with `windows_vdvs_version` (below) to generate a URL of the form <windows_vdvs_path>_<windows_vdvs_version>.zip to download the software. For example, to download version 0.21, set `windows_vdvs_path` equal to `https://vmware.bintray.com/vDVS/vsphere-storage-for-docker_windows` and `windows_vdvs_version` equal to `0.21`
windows_vdvs_version	group_vars/vars	Combined with `windows_vdvs_path`, this variable is used to generate the URL for downloading the software.
windows_vdvs_directory	group_vars/vars	Variable used to determine where vDVS software will be unzipped and installed from. The default is `C:\Users\Administrator\Downloads`

group_vars/win_worker.yml

There is a separate file in the `group_vars` directory named `win_worker.yml` for advanced Windows-specific configuration. These variables are used in the following playbooks:

- playbooks/create_windows_vms.yml
- playbooks/install_docker_window.yml
- playbooks/scale_workers_windows.yml

In general, it should not be necessary to modify this file, but the variables are documented here for the sake of completeness.

Table 14. Advanced windows variables

Variable	File	Description
ansible_user	**group_vars/win_worker.yml**	Defaults to the Windows user account `win_username` as specified in `groupr_vars/vars`
ansible_password	**group_vars/win_worker.yml**	Defaults to the Windows user password `win_password` as specified in `group_vars/vault`
ansible_port	**group_vars/win_worker.yml**	5986
ansible_connection	**group_vars/win_worker.yml**	winrm
ansible_winrm_server_cert_validation	**group_vars/win_worker.yml**	Defaults to `ignore`
ansible_winrm_operation_timeout_sec	**group_vars/win_worker.yml**	Defaults to `250`
ansible_winrm_read_timeout_sec	**group_vars/win_worker.yml**	Defaults to `300`
windows_timezone	**group_vars/win_worker.yml**	Defaults to `15`

Splunk configuration

This solution supports two types of Splunk deployment. Firstly, there is a built-in deployment useful for demos and for getting up to speed with Splunk. Alternatively, the solution can be configured to interact with a standalone, production Splunk deployment that you set up independently. In this case, you must explicitly configure the universal forwarders with external "forward servers" (Splunk indexers), whereas this happens automatically with the built-in option.

In the standalone deployment, you can enable SSL authentication between the universal forwarders and the indexers, by setting the `splunk_ssl` variable to `yes` in the file `group_vars/vars`. The built-in deployment does not support SSL and so, in this instance, the value of the `splunk_ssl` variable is ignored. For more information on enabling SSL, see Appendix C.

After the installation is complete, the Splunk UI can be reached at `http://<fqdn>:8000`, where `<fqdn>` is the FQDN of one of your Linux Docker nodes. Mesh routing does not currently work on Windows so you must use a Linux node to access the UI.

Splunk prerequisites

You should select the Splunk deployment type that you require by setting the variable `monitoring_stack` in the `group_vars/vars` file to either **splunk**, to use a standalone Splunk deployment, or **splunk_demo** for the built-in version. If you omit this variable, or if it has an invalid value, no Splunk deployment will be configured.

For both types of deployment, you need to download the Splunk Universal forwarder images/packages from https://www.splunk.com/en_us/download/universal-forwarder.html. Packages are available for 64-bit Linux and 64-bit Windows 8.1/Windows 10. Download the RPM package for Linux 64-bit (2.6+ kernel Linux distributions) to `./files/splunk/linux`. If you are deploying Windows nodes, download the MSI package for Windows 64 bit to `./files/splunk/windows`. For a dual Linux/Windows deployment, the images and packages must have same name and version, along with the appropriate extensions, for example:

- files/splunk/windows/splunkforwarder-7.0.2.msi
- files/splunk/linux/splunkforwarder-7.0.2.rpm

You need to set the variable `splunk_architecture_universal_forwarder_package` to the name you selected for the package(s), not including the file extension. Depending on the Splunk deployment you have chosen, edit the file `templates/monitoring/`**splunk**`/vars.yml` or the file `templates/monitoring/`**splunk_demo**`/vars.yml` and set the variable, for example:

```
splunk_architecture_universal_forwarder_package: 'splunkforwarder-7.0.2'
```

If you are using a standalone Splunk deployment, you must specify the list of indexers using the variable `splunk_architecture_forward_servers` in `group_vars/vars`, for example:

```
splunk_architecture_forward_servers:
- splunk-indexer1.cloudra.local:9997
- splunk-indexer2.cloudra.local:9997
```

By default, the indexers are configured in a single load balancing group. This can be changed by editing the file `outputs.conf.j2` in the folder `template/monitoring/splunk/`. For more information on forwarding using Universal Forwarder, see the Splunk documentation at http://docs.splunk.com/Documentation/Forwarder/7.0.2/Forwarder/Configureforwardingwithoutputs.conf.

On your standalone Splunk installation, you need to install the following add-ons and apps.

To monitor **Linux Worker nodes**, the **Docker app** should be installed on central Splunk. More info on this Docker app can be found at https://github.com/splunk/docker-itmonitoring and at https://hub.docker.com/r/splunk/universalforwarder/.

To monitor the **Windows worker nodes**, install the **Splunk App for Windows Infrastructure** on central Splunk and its dependencies:

- Splunk App for Windows Infrastructure - see https://splunkbase.splunk.com/app/1680/

- Splunk Add-on for Microsoft Windows - see https://splunkbase.splunk.com/app/742/
- Splunk Add-on for Microsoft Windows DNS (if this is not installed on central Splunk, you will see yellow icons on some dashboards with the message `eventtype wineventlog-dns does not exist or is disabled`) - see https://splunkbase.splunk.com/app/3208/
- Splunk Supporting Add-on for Active Directory (if this is not installed on central Splunk, you will see yellow icons on some dashboards with the message `eventtype wineventlog-ds does not exist or is disabled`) - see https://splunkbase.splunk.com/app/1151/

If you want to use your own certificates in your standalone Splunk deployment to secure the communications between the indexers and the universal forwarders, see the section Enabling SSL.

You can specify advanced Splunk configuration in the following files:

- files/splunk/linux/SPLUNK_HOME
- files/splunk/linux/DOCKER_TAS
- files/splunk/windows/SPLUNK_HOME

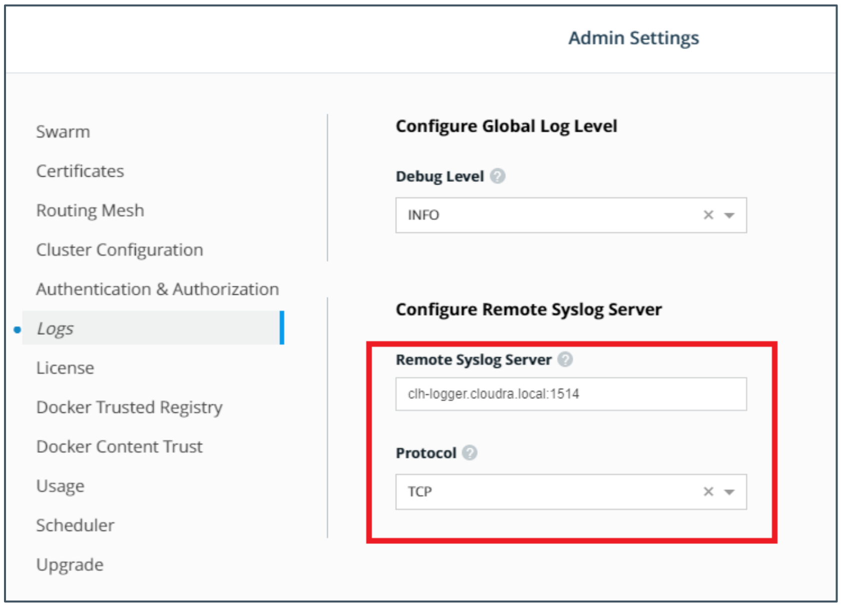
These files will be copied as-is to the systems running the universal forwarder.

Configuring syslog in UCP

In order to see some data in the UCP operational dashboard, you need to have UCP send its logs to the VM configured in the [logger] group. For example, for the following `vm_host` file:

```
[logger]
hpe-logger ip_addr='10.60.59.24/16' esxi_host='esxi-hpe-2.cloudra.local'
```

This will configure UCP to send its logs to `hpe-logger.cloudra.local:1514`. You need to select the TCP protocol as shown in Figure 7.



Configure Global Log Level

Debug Level: INFO

Configure Remote Syslog Server

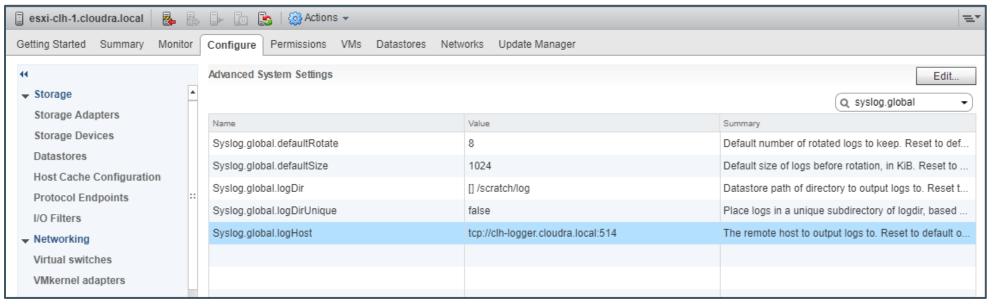
Remote Syslog Server: clh-logger.cloudra.local:1514

Protocol: TCP

Figure 7. Configure Remote Syslog Server in UCP

Configuring syslog in ESX

This configuration must be done manually for each ESX server. The syslog server should be the server configured in the [logger] group in your `vm_hosts` inventory. The protocol should be `tcp` and the port `514` as shown in Figure 8.



Advanced System Settings

syslog.global

Name	Value	Summary
Syslog.global.defaultRotate	8	Default number of rotated logs to keep. Reset to def...
Syslog.global.defaultSize	1024	Default size of logs before rotation, in KiB. Reset to ...
Syslog.global.logDir	[] /scratch/log	Datastore path of directory to output logs to. Reset t...
Syslog.global.logDirUnique	false	Place logs in a unique subdirectory of logdir, based ...
Syslog.global.logHost	tcp://clh-logger.cloudra.local:514	The remote host to output logs to. Reset to default o...

Figure 8**.** Configure Syslog on ESXi Hosts

For more information, see the VMware documentation at https://docs.vmware.com/en/VMware-vSphere/6.5/com.vmware.vsphere.security.doc/GUID-9F67DB52-F469-451F-B6C8-DAE8D95976E7.html.

Related playbooks

- `playbooks/monitoring.yml` installs and configures the Splunk Universal Forwarders
- `playbooks/splunk_demo.yml` installs a demo of Splunk Enterprise in the cluster (if the `splunk_demo` deployment option is selected)

Limitations

- The Docker App has a number of open issues
  - https://github.com/splunk/docker-itmonitoring/issues/19
  - https://github.com/splunk/docker-itmonitoring/issues/20
- The Docker events tab is not working

Sysdig configuration

The playbook `playbooks/install_sysdig.yml` is used to automate the configuration of the SaaS setup. By default, this playbook is commented out in `site.yml` and must be explicitly enabled. An access key variable must be set in the `group_vars/vault` file as detailed in Table 15.

Table 15. Sysdig variables

Variable	File	Description
sysdig_access_key	**group_vars/vault**	After the activation of your account on the Sysdig portal, you will be provided with your access key which will be used by the playbooks to install the agent on each UCP, DTR and Linux worker node, as well as the NFS, logger and load balancer VMs.
sysdig_agent	group_vars/vars	Specifies the URL to the Sysdig Linux native install agent, for example, `https://s3.amazonaws.com/download.draios.com/stable/install-agent`
sysdig_tags	group_vars/vars	Tagging your hosts is highly recommended. Tags allow you to sort the nodes of your infrastructure into custom groups in Sysdig Monitor. Specify location, role, and owner in the format: `'location:City,role:Enterprise CaaS on Synergy,owner:Customer Name'`

Note

By default, you must have outgoing port 6666 open in your firewall, to allow data to flow to `collector.sysdigcloud.com`. You can configure the agent to use a different port by setting the `collector_port` parameter in the configuration file `/opt/draios/etc/dragent.yaml`. For more information, see the relevant Sysdig documentation at https://support.sysdig.com/hc/en-us/articles/204205969.

If you are using a proxy, it must be configured to be "fully-transparent". Non-transparent proxies will not allow the agent to connect.

Using the Sysdig software as a solution (SaaS) website https://app.sysdigcloud.com, you are able to view, analyze and inspect various different dashboards.

Prometheus and Grafana configuration

All Monitoring-related variables for Prometheus and Grafana are described in Table 16. The variables determine the versions of various software tools that are used and it is recommended that the values given below are used.

Table 16. Monitoring variables

Variable	Description
cadvisor_version	`v0.25.0`
node_exporter_version	`v1.14.0`
prometheus_version	`v1.7.1`
grafana_version	`4.4.3`
prom_persistent_vol_name	The name of the volume which will be used to store the monitoring data. The volume is created using the vsphere docker volume plugin.
prom_persistent_vol_size	The size of the volume which will hold the monitoring data. The exact syntax is dictated by the vSphere Docker Volume plugin. The default value is 10GB.

Protecting sensitive information

A vault file is used to protect any sensitive variables that should not appear in clear text in your `group_vars/vars` file. The vault file will be encrypted and will require a password to be entered before it can be read or updated.

A sample vault file is provided 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 called `group_vars/vault` and add entries similar to:

```
---
docker_ee_url: 'your_url_here'
vcenter_password: 'xxxx'
vm_password: 'xxxx'
ucp_password: 'zzzz'
win_password: 'yourpass'
sysdig_access_key: 'enter_sysdig_access_key'
rhn_orgid: "YourOrgId"
rhn_key: "YourActivationKey"
```

`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 avoid illegitimate 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 using a Git repository to manage your playbooks.

Overview of the playbooks

site.yml and related playbooks

The playbook `./site.yml` is the day 0 playbook you use to deploy the solution. It is the main entry point and invokes the playbooks described in this section.

- `playbooks/create_vms.yml` will create all the necessary Virtual Machines for the environment from the VM Template defined in the `vm_template` variable.
- `playbooks/config_networking.yml` will configure the network settings in all the Virtual Machines.
- `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.
- `playbooks/install_haproxy.yml` installs and configures the HAProxy package in the load balancer nodes. HAProxy is the tool chosen to implement load balancing for UCP nodes, DTR nodes and worker nodes.
- `playbooks/config_ntp.yml` configures the **chrony** client package in all Virtual Machines in order to have a synchronized clock across the environment. It will use the list of servers specified in the `ntp_servers` variable in the file `group_vars/vars`.
- `playbooks/install_docker.yml` installs Docker along with all its dependencies.
- `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.
- `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.
- `playbooks/docker_post_config.yml` performs a variety of tasks to complete the installation of the Docker environment, including configuration of the HTTP/HTTPS proxies, if any, and installation of the VMware vSphere Storage for Docker volume plugin.
- `playbooks/install_nfs_server.yml` installs and configures an NFS server on the NFS node.
- `playbooks/install_nfs_clients.yml` installs the required packages on the DTR nodes to be able to mount an NFS share.
- `playbooks/create_main_ucp.yml` installs and configures the first Docker UCP instance on the target node defined by the group `ucp_main` in the `vm_hosts` inventory.
- `playbooks/scale_ucp.yml` installs and configures additional instances of UCP on the target node defined by the group `ucp` in the vm_hosts inventory, except for the node defined in the group `ucp_main`.
- `playbooks/create_main_dtr.yml` installs and configures the first Docker DTR instance on the target nodes defined by the group `dtr_main` in the `vm_hosts` inventory.
- `playbooks/scale_workers.yml` installs and configures additional workers on the target nodes defined by the group `worker` in the `vm_hosts` inventory.

- `playbooks/install_logspout.yml` installs and configures **Logspout** on all Docker nodes. Logspout is responsible for sending logs produced by containers running on the Docker nodes to the central logger VM. By default, this playbook is commented out in `site.yml`.
- `playbooks/config_monitoring.yml` configures a monitoring system for the Docker environment based on Grafana, Prometheus, cAdvisor and node-exporter Docker containers. By default, this playbook is commented out in `site.yml`, so if you want to use the solution to automatically deploy a Prometheus/Grafana monitoring system, you must explicitly uncomment both this and the `playbooks/install_logspout.yml` playbook.
- `playbooks/monitoring.yml` installs and configures the Splunk Universal Forwarders. The variable `monitoring_stack` in `group_vars/vars` is used to specify the type of deployment you want. A value of `splunk_demo` will result in this playbook deploying a Splunk enterprise instance for demo purposes. A value of `splunk` is used to configure an external production Splunk deployment.
- `playbooks/splunk_demo.yml` installs a demo of Splunk Enterprise in the cluster (if the `splunk_demo` deployment option is selected)
- `playbooks/config_scheduler.yml` configures the scheduler to prevent regular users (i.e. non-admin users) to schedule containers on the Docker nodes running instances of UCP and DTR.
- `playbooks/scale_dtr.yml` installs and configures additional instances (or replicas) of DTR on the target nodes defined by the group `dtr` in the `vm_hosts` inventory, with the exception of the node defined in the group `dtr_main`.
- `playbooks/reconfigure_dtr.yml` is used to reconfigure DTR with the FQDN of the UCP Load Balancer and also enables image scanning.
- `playbooks/install_sysdig.yml` is used to configure Sysdig. It opens the required port in the firewall, and installs the latest version of the Sysdig agent image on the nodes. By default, this playbook is commented out in `site.yml`, so if you want to use the solution to automatically configure Sysdig, you must uncomment this line.

Windows playbooks

- `playbooks/create_windows_vms.yml` will create all the necessary Windows 2016 VMs for the environment based on the Windows VM Template defined in the `win_vm_template` variable.
- `playbooks/install_docker_windows.yml` installs Docker along with all its dependencies on your Windows VMs
- `playbooks/scale_workers_win.yml` installs and configures additional Windows workers on the target nodes defined by the group `win_worker` in the `vm_hosts` inventory.

Backup and restore playbooks

Backup playbooks

- `playbooks/backup_swarm.yml` is used to back up the swarm data
- `playbooks/backup_ucp.yml` is used to back up UCP
- `playbooks/backup_dtr_meta.yml` is used to back up DTR metadata
- `playbooks/backup_dtr_images.yml` is used to back up DTR images

Restore playbooks

- `playbooks/restore_dtr_images.yml` is used to restore DTR images
- `playbooks/restore_dtr_metadata.yml` is used to restore DTR metadata
- `playbooks/restore_ucp.yml` is used to restore UCP

Convenience playbooks

- `playbooks/clean_all.yml` powers off and deletes all VMs in your inventory.
- `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 not run from `site.yml` by default.

Convenience scripts

- `backup.sh` can be used to take a backup of the swarm, UCP DTR and the DTR images in one go.
- `restore_dtr.sh` can be used to restore DTR metadat and DTR images.
- `scale_worker.sh` can be used to scale the worker nodes.

Running the playbooks

At this point, the system is ready to be deployed. Make sure you are logged on as root in your ansible box and that your current directory is `/root/Docker-Synergy/ops`

To start a Linux-only deployment, use the following command:

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

The playbooks should run for 35-40 minutes depending on your server specifications and the size of your environment.

To start a hybrid, Windows and Linux deployment, use the following command:

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

The playbooks should run for 70-80 minutes depending on your server specifications and the size of your environment. The increase in running time is primarily due to the need to update Windows after creating the VMs.

Post deployment

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

The ansible log is stored in the `ops` folder of the repo (`/root/Docker-Synergy/ops`). If the deployment fails, you may find useful hints in this log.

To see how to check if the certs have been deployed correctly, see Appendix D.

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 life cycle management of the different components that are used in this solution.

The lifecycle of the following stacks need to be maintained and managed.

1. Monitoring Tools (Splunk or Prometheus and Grafana)
2. Docker Enterprise Edition Environment
3. Virtual Machine Operating systems
4. Synergy environment

The general practice and recommendation is to follow a bottom-up approach for updating all components of the environment and making sure the dependencies are met. In our solution, we would start with Synergy and end with the monitoring environment. If all components are not being updated at the same time, the same approach can be followed – updating only the components that require updates while adhering to the interdependencies of each component that is being updated.

Synergy environment

HPE Synergy Composer powered by HPE OneView provides fast, reliable, and simplified firmware and driver management across many HPE Synergy components. HPE OneView manages firmware to reduce manual interactions and errors, in addition to minimizing downtime. Firmware updates of management appliances and shared infrastructure are non-disruptive to the production workload.

More information is available at Best Practices for HPE Synergy Firmware and Driver Updates

VMware Components

The solution in this deployment guide is built on VMware vSphere and leverages VMware ESXi and vCenter. For more information on upgrading vSphere, see the VMware documentation: Introduction to vSphere Upgrade.

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 17). 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 17. vSphere Docker Volume service components

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

Red Hat Enterprise Linux operating system

This solution is built using Red Hat Enterprise Linux (see Table 18) 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.

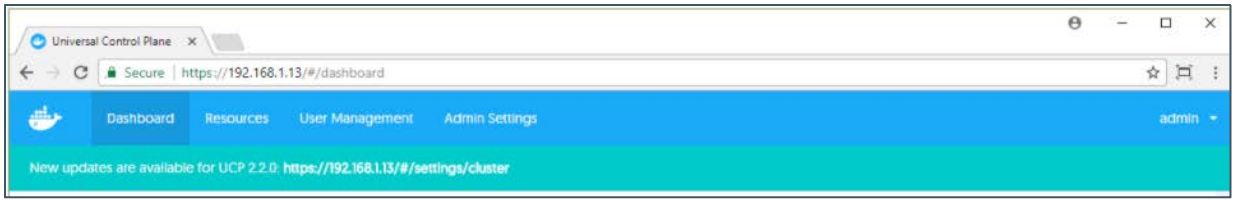
Table 18. Operating system

Order	Component	Dependency (compatibility)	Download/Documentation
1.	Red Hat Enterprise Linux	Docker EE 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 Engine. It is imperative that the components belonging to the same version are deployed or upgraded together – see Table 19.

A banner will be displayed on the UI, as shown in Figure 13, when an update is available for UCP or DTR. You can start the upgrade process by clicking on the banner.



New updates are available for UCP 2.2.0: https://192.168.1.13/#/settings/cluster

Figure 9**.** Docker update notification

Table 19. Docker EE components

Order	Component	Dependency (compatibility)	Download/Documentation
1.	Docker Daemon/Engine	VM Operating System	Docker Lifecycle Maintenance
2.	Universal Control Plane	vDVS plugin	Docker Compatibility Matrix
3.	Docker Trusted Registry	Prometheus and Grafana	

Monitoring Tools

To learn more about upgrading Splunk, see the relevant documentation at How to upgrade Splunk Enterprise

The Sysdig agent runs as a container and the latest version is pulled from the Docker hub on first installation. Re-run the `install_sysdig.yml` playbook to update to the newest version if required.

Prometheus and Grafana monitoring tools (see Table 20) run as containers within the Docker environment. Newer versions of these tools can be deployed by pulling the Docker images from Docker Hub. Verify that the version of Prometheus that is being used is compatible with the version of Docker EE.

Table 20. Monitoring tools: Prometheus and Grafana

Order	Component	Dependency (compatibility)	Download/Documentation
1.	Prometheus	Grafana Docker EE	Prometheus Images on Docker Hub Upgrading Grafana
2.	Grafana	Prometheus Docker EE	

Windows operating system and Docker EE

Docker Enterprise Edition for Windows Server (Docker EE) enables native Docker containers on Windows Server. Windows Server 2016 and later versions are supported. This solution has been tested with Windows worker nodes running Windows Server 2016 and with Docker EE 17.06.

Note

Docker Universal Control Plane is not currently supported on Windows Server 1709 due to image incompatibility issues. For more information, see the Docker documentation Install Docker Enterprise Edition for Windows Server

This solution recommends that you only run Windows Server 2016 on your Windows worker nodes and that you install any required updates to your Windows nodes in a timely manner.

For information on how to update Docker EE on Windows Server 2016, see the Docker documentation Update Docker EE

Backup and restore

Backup and restore UCP and DTR

The playbooks provided in this solution implement the backup and restore procedures as they are described in the Docker documentation at https://docs.docker.com/enterprise/backup/. The solution follows the recommendations in the Docker best practices document at https://success.docker.com/article/backup-restore-best-practices.

Note

It is important that you make copies of the backed up data and that you store the copies in a separate physical location. You must also recognize that the backed up data contains sensitive information such as private keys and so it is important to restrict access to the generated files. However, the playbooks do not backup the sensitive information in your `group_vars/vault` file so you should make sure to keep track of the credentials for the UCP Administrator.

Warning

The restore procedures do not restore swarm data. You should follow infrastructure as code (IaC) guidelines and maintain your service, stack and network definitions using source code or configuration management tools. You must also ensure that you safely manage the credentials of your administration accounts as existing UCP Client bundles will not work when you restore UCP on a new swarm.

Backup UCP and DTR

Backup configuration variables

Table 21 shows the variables related to backing up UCP and DTR. All these variables are defined in the file **group_vars/backup**. All the data that is backed up is streamed over an SSH connection to the backup server. Currently, the playbooks only support the use of the Ansible box as the backup server.

Table 21. Backup variables

Variable	File	Description
backup_server	**group_vars/backup**	Currently, the playbooks only support the use of the Ansible box as the backup server.
backup_dest	**group_vars/backup**	This variable should point to an existing folder on your ansible box where the `root` user has write access. All the backups will be stored in this folder. For example, `/root/backup`
#swarm_offline_backup	**group_vars/backup**	This variable is commented out by default. More information on this variable is provided below.

Backing up the Swarm

When you back up the swarm, your services and stack definitions are backed up together with the networks definitions. However, Docker volumes or their contents will not be backed up. (If Docker volumes are defined in stacks, they will be re-created when you restore the stacks, but their content will be lost). You can back up the swarm using the playbook named `backup_swarm.yml` which is located in the `playbooks` folder on your Ansible server. The playbook is invoked as follows:

```
# ansible-playbook -i vm_hosts playbooks/backup_swarm.yml
```

This playbook creates two archives in the folder specified by the variable `backup_dest` in `group_vars/backup`. By default, the file is named using the following pattern:

```
<backup_dest>/backup_swarm_<vmname>_<timestamp>.tgz
<backup_dest>/backup_swarm_<vmname>_<timestamp>.vars.tgz
```

`<vmname>` is the name of the host (in the inventory) that was used to take the backup, and `<timestamp>` is the time at which the backup was taken. The file with the extension `.vars.tgz` contains information regarding the system that was backed up.

You can override the generated file name by defining the variable **backup_name** on the command line when running the playbook. In the example below:

```
# ansible-playbook -i vm_hosts playbooks/backup_swarm.yml -e backup_name=my_swarm_backup
```

The generated files won't have `<vmname>` or `<timestamp>` appended:

```
<backup_dest>/my_swarm_backup.tgz
<backup_dest>/my_swarm_backup.vars.tgz
```

Warning
Online versus offline backups: By default, the playbook performs online backups. You can take offline backups by setting the variable `swarm_backup_offline` to "`true`". The playbook will then stop the Docker daemon on the machine used to take the backup (a manager/UCP node). Before it does so, the playbook will verify that enough managers are running in the cluster to maintain the quorum. If this is not the case, the playbook will exit with an error. For more information, see the Docker documentation at https://docs.docker.com/engine/swarm/admin_guide/#recover-from-disasterv

Backing up the Universal Control Plane (UCP)

When you backup UCP, you save the following data/metadata:

Table 22. UCP data backed up

Data	Description
Configurations	The UCP cluster configurations, as shown by `docker config ls`, including Docker EE license and swarm and client CAs
Access control	Permissions for team access to swarm resources, including collections, grants, and roles
Certificates and keys	The certificates, public keys, and private keys that are used for authentication and mutual TLS communication
Metrics data	Monitoring data gathered by UCP
Organizations	Your users, teams, and orgs
Volumes	All UCP named volumes, which include all UCP component certs and data

To make a backup of UCP, use `playbook/backup_ucp.yml` as follows:

```
# ansible-playbook -i vm_host playbooks/backup_ucp.yml
```

This playbook creates two archives in the folder specified by the variable `backup_dest` in `group_vars/backup`. By default, the files are named using the following pattern:

```
<backup_dest>/backup_ucp_<ucpid>_<vmname>_<timestamp>.tgz
<backup_dest>/backup_ucp_<ucpid>_<vmname>_<timestamp>.vars.tgz
```

`<ucpid>` is the ID of the UCP instance, `<vmname>` is the name of the host (in the inventory) that was used to take the backup, and `<timestamp>` is the time at which the backup was taken. The file with the extension `.vars.tgz` contains information regarding the system which was backed up.

You can override the generated file name by defining the variable **backup_name** on the command line when running the playbook. In the example below:

```
# ansible-playbook -i vm_hosts playbooks/backup_ucp.yml -e backup_name=my_ucp_backup
```

The generated files won't have `<vmname>` or `<timestamp>` appended:

```
<backup_dest>/my_ucp_backup.tgz
<backup_dest>/my_ucp_backup.vars.tgz
```

Warning
To create a consistent backup, the backup command **temporarily stops the UCP containers running on the node where the backup is being performed**. User resources, such as services, containers, and stacks are not affected by this operation and will continue to operate as expected. Any long-lasting `docker exec`, `docker logs`, `docker events`, or `docker attach` operations on the affected manager node will be disconnected.

For more information on UCP backup, see the Docker documentation at https://docs.docker.com/datacenter/ucp/2.2/guides/admin/backups-and-disaster-recovery/

Backing up the Docker Trusted Registry (DTR)

When you backup DTR, you save the following data/metadata:

Table 23. UCP data backed up

Data	Backed up?	Description
Configurations	yes	DTR settings
Repository metadata	yes	Metadata like image architecture and size
Access control to repos and images	yes	Data about who has access to which images
Notary data	yes	Signatures and digests for images that are signed
Scan results	yes	Information about vulnerabilities in your images
Certificates and keys	yes	TLS certificates and keys used by DTR
Image content	no	Needs to be backed up separately, depends on DTR configuration
Users, orgs, teams	no	Create a UCP backup to backup this data
Vulnerability database	no	Can be re-downloaded after a restore

To make a backup of DTR metadata, use `playbook/backup_dtr_metadata.yml`

```
# ansible-playbook -i vm_host playbooks/backup_dtr_metadata.yml
```

This playbook creates two archives in the folder specified by the variable `backup_dest` in `group_vars/backup`. By default, the file is named using the following pattern:

```
<backup_dest>/backup_dtr_meta_<replica_id>_<vmname>_<timestamp>.tgz
<backup_dest>/backup_dtr_meta_<replica_id>_<vmname>_<timestamp>.vars.tgz
```

`<replica_id>` is the ID of the DTR replica that was backed up, `<vmname>` is the name of the host (in the inventory) that was used to take the backup, and `<timestamp>` is the time at which the backup was taken. The file with the extension `.vars.tgz` contains information regarding the system that was backed up.

You can override the generated file name by defining the variable **backup_name** on the command line when running the playbook. In the example below:

```
# ansible-playbook -i vm_hosts playbooks/backup_dtr_metadata.yml -
e backup_name=my_dtr_metadata_backup
```

The generated files won't have `<vmname>` or `<timestamp>` appended:

```
<backup_dest>/my_dtr_metadata_backup.tgz
<backup_dest>/my_dtr_metadata_backup.vars.tgz
```

For more information on DTR backups, see the Docker documentation at https://docs.docker.com/datacenter/dtr/2.4/guides/admin/backups-and-disaster-recovery/

Backing up DTR data (images)

To make a backup of the images that are inventoried in DTR and stored on the NFS server, use `playbooks/backup_dtr_images.yml`

```
# ansible-playbook -i vm_host playbooks/backup_dtr_images.yml
```

This playbook creates two archives in the folder specified by the variable `backup_dest` in `group_vars/backup`. By default, the file is named using the following pattern:

```
<backup_dest>/backup_dtr_data_<replica_id>_<vmname>_<timestamp>.tgz
<backup_dest>/backup_dtr_data_<replica_id>_<vmname>_<timestamp>.vars.tgz
```

`<replica_id>` is the ID of the DTR replica that was backed up, `<vmname>` is the name of the host (in the inventory) that was used to take the backup, and `<timestamp>` is the time at which the backup was taken.

You can override the generated file name by defining the variable **backup_name** on the command line when running the playbook, as shown in the example below:

```
# ansible-playbook -i vm_hosts playbooks/backup_dtr_images.yml -
e backup_name=my_dtr_data_backup
```

The generated file won't have `<vmname>` or `<timestamp>` appended:

```
<backup_dest>/my_dtr_data_backup.tgz
<backup_dest>/my_dtr_data_backup.vars.tgz
```

For more information on DTR backups, see the Docker documentation at
https://docs.docker.com/datacenter/dtr/2.4/guides/admin/backups-and-disaster-recovery/

Backing up other metadata, including passwords

The backup playbooks do not backup the sensitive data in your `group_vars/vault` file. The information stored in the `.vars.tgz` files includes backups of the following files:

- **vm_hosts**, a copy of the `vm_hosts` file at the time the backup was taken
- **vars**, a copy of your `group_vars/vars` file at the time the backup was taken
- **meta.yml**, a generated file containing information pertaining to the backup

The **meta.yml** file contains the following information:

```
backup_node=”<node that took the backup>”
replica_id=”<ID of DTR replica if DTR backup>”
backup_source=””
ucp_version=”<UCP version if UCP backup>”
dtr_version=”<DTR version of DTR backup>”
```

Backup Utility

The script `backup.sh` can be used to take a backup of the swarm, UCP DTR and the DTR images in one go. You can pass this script an argument (tag) that will be used to prefix the backup filenames, thereby overriding the default naming. The following table shows the file names produced by `backup.sh` based on the argument passed in the command line.

Table 24. Backup utility

Example	Command line	Generated filenames
Default	`./backup.sh`	backup_swarm_<vmname>_<timestamp>.tgz, backup_ucp_<ucpid>_<vmname>_<timestamp>.tgz, backup_dtr_meta_<replica_id>_<vmname>_<timestamp>.tgz, backup_dtr_data_<replica_id>_<vmname>_<timestamp>.tgz and the corresponding `.vars.tgz` files

Custom	`./backup.sh my_backup`	my_backup_swarm.tgz, my_backup_ucp.tgz, my_backup_dtr_meta.tgz, my_backup_dtr_data.tgz, and the corresponding `.vars.tgz` files
Date	`./backup.sh $(date '+%Y_%m_%d_%H%M%S')`	<date>_swarm.tgz, <date>_ucp.tgz, <date>_dtr_meta.tgz, <date>_dtr_data.tgz, and the corresponding `.vars.tgz` files

In addition, the `backup.sh` script accepts an optional switch that will let you specify the location of the password file that will be passed to the `ansible-playbook` commands in the script. This is required if you encrypted the `group_vars/vault` file. The general syntax for this script is as follows:

```
./backup.sh [ -v <Vault Password File> ] [ tag ]
```

Related playbooks

- `playbooks/backup_swarm.yml` is used to back up the swarm data
- `playbooks/backup_ucp.yml` is used to back up UCP
- `playbooks/backup_dtr_meta.yml` is used to back up DTR metadata
- `playbooks/backup_dtr_images.yml` is used to back up DTR images

Restoring your cluster after a disaster

The playbooks address a disaster recovery scenario where you have lost your entire cluster and all the VMs. Other scenarios and how to handle them are described in the Docker documentation including the following scenarios:

- You have lost one UCP instance but your cluster still has the quorum. The easiest way is to recreate the missing UCP instance from scratch.
- You have lost the quorum in your UCP cluster but there is still one UCP instance running.
- You have lost one instance of DTR but still have a quorum of replicas. The easiest way is to recreate the missing DTR instance from scratch.
- You have lost the quorum of your DTR cluster but still have one DTR instance running.

Before you restore

Step 1. Retrieve the backup files using your chosen backup solution and save them to a folder on your Ansible server. If you have used timestamps in the naming of your backup files, you can use them to determine the chronological order. If you used the `backup.sh` script specifying a date prefix, you can use that to identify the matching set of backup files. You should choose the files in the following reverse chronological order, from the most recent to the oldest file. Make sure you restore both the *.tgz and the *.vars.tgz files.

1. DTR images backup
2. DTR metadata backup
3. UCP backup
4. Swarm backup

In this example, we will assume a set of backup files stored in `/root/restore` that were created specifying a date prefix. These will have names like `2018_04_17_151734_swarm.tgz`, `2018_04_17_151734_ucp.tgz`, etc and the corresponding `.vars.tgz` files.

Step 2: Retrieve the DTR replica ID, the DTR version and the UCP version

To retrieve the ID of the replica that was backed up, as well as the version of DTR, you need to extract the data from the `.vars.tgz` file associated with the archive of the DTR metadata. You can retrieve this as follows:

```
[root@hpe2-ansible ops]# tar -
Oxf /root/restore/2018_04_17_151734_dtr_meta.vars.tgz meta.yml
```

```
backup_node="hpe-dtr01"
replica_id="ad5204e8a4d0"
backup_source=""
ucp_version=""
dtr_version="2.4.3"


[root@hpe2-ansible ops]# tar -Oxf /root/restore/2018_04_17_151734_ucp.vars.tgz meta.yml
backup_node="hpe-ucp01"
replica_id=""
backup_source=""
ucp_version="2.2.7"
dtr_version=""
```

Take note of the replica ID (ad5204e8a4d0), the version of DTR (2.4.3) and the version of UCP (2.2.7).

Step 3: Populate the `group_vars/backups` file

```
backup_swarm: "/root/restore/2018_04_17_151734_swarm.tgz"
backup_ucp: "/root/restore/2018_04_17_151734_ucp.tgz"
backup_dtr_meta: "/root/restore/2018_04_17_151734_dtr_meta.tgz"
backup_dtr_data: "/root/restore/2018_04_17_151734_dtr_data.tgz"
backup_dtr_id: "ad5204e8a4d0"
backup_dest: "/root/backups"
backup_server: <IP of your ansible box>
```

You should populate your `group_vars/backups` file as above, with the `backup_dtr_id` variable containing the value you retrieved in the preceding step as `replica_id=`"**ad5204e8a4d0**".

Step 4: Verify that your `group_vars/vars` file specifies the correct versions of DTR and UCP.

The playbooks use the versions of UCP and DTR as specified in your `group_vars/vars` file to restore your backups. You must ensure that the versions specified in your current `group_vars/vars` file correspond to the versions in the backups as determined above.

```
[root@hpe2-ansible ops]# cat group_vars/vars | grep dtr_version
dtr_version: '2.4.3'


[root@hpe2-ansible ops]# cat group_vars/vars | grep ucp_version
ucp_version: '2.2.7'
```

Step 5: Restore UCP admin credentials if required

You must ensure that the UCP admin credentials in your current `group_vars/vars` file are those that were in effect when you generated the backup files. If they have changed since then, you must restore the original credentials for the duration of the restore procedure.

Step 6: Restore your inventory (`vm_hosts`)

Your inventory must reflect the environment that was present when the backup files were created. You can find a copy of the inventory as it was when the backup was taken in the `*.vars.tgz` files.

Restore UCP and DTR

Warning
This procedure is aimed at restoring a cluster after a disaster. It assumes you have lost all the VMs in your cluster and want to redeploy using data that you backed up earlier. The solution follows Docker best practice, which means the swarm artifacts are not restored. You will need to restore your Docker volumes and your applications (stacks and services) when this procedure is complete.

---

1. Ensure that you have completed all the preliminary steps as outlined in the section Before you restore
2. Run the restore playbook

```
ansible-playbook -i vm_hosts restore.yml
```

3. If you are using the image scanning functionality in DTR, you will need to re-download the vulnerability database. For more information, see the Docker documentation here.

You are now ready to restore your Docker volumes and your applications.

Restore DTR metadata and DTR images

Note
This procedure restores DTR metadata and images and assumes you have lost all the DTR VMs in your cluster. It will redeploy using the DTR data that you backed up earlier and will also restore the images if the folder exported by the NFS VM is empty.

---

1. Ensure that you have completed all the preliminary steps as outlined in the section Before you restore. In this scenario, you need the archives for the DTR metadata and the DTR images.
2. Ensure that all the DTR VMs listed in your inventory are destroyed, using the vSphere Web Client to delete them if required. If you want to restore the DTR images you should also delete the NFS VM.
3. Remove the DTR nodes from the swarm by running the `docker node rm <DTR node>` command on a UCP node for each DTR node in your cluster. The following example shows the sequence of commands to use to remove the DTR nodes:

```
[root@hpe2-ansible ops]# docker node ls
ID       HOSTNAME                     STATUS             AVAILABILITY
aiz… *   hpe-ucp02.cloudra.local      Ready              Active
gvf…     hpe-dtr01.cloudra.local      Down               Active
ir4…     hpe-ucp03.cloudra.local      Ready              Active
mwf…     hpe-dtr02.cloudra.local      Down               Active
oqy…     hpe-ucp01.cloudra.local      Ready              Active
xqe…     hpe-worker01.cloudra.local   Ready              Active
zdu…     hpe-dtr03.cloudra.local      Down               Active


[root@hpe2-ansible ops]# docker node rm hpe-dtr01.cloudra.local
hpe-dtr01.cloudra.local
[root@hpe2-ansible ops]# docker node rm hpe-dtr02.cloudra.local
hpe-dtr02.cloudra.local
[root@hpe2-ansible ops]# docker node rm hpe-dtr03.cloudra.local
hpe-dtr03.cloudra.local
```

```
[root@hpe2-ansible ops]# docker node ls
ID        HOSTNAME                     STATUS              AVAILABILITY
aiz…      hpe-ucp02.cloudra.local      Ready               Active
ir4…      hpe-ucp03.cloudra.local      Ready               Active
oqy… *    hpe-ucp01.cloudra.local      Ready               Active
xqe…      hpe-worker01.cloudra.local   Ready               Active
```

4. Run the restore script:

   ```
   ./restore_dtr.sh
   ```

5. If you are using the image scanning functionality in DTR, you will need to re-download the vulnerability database. For more information, see the Docker documentation here.

Related playbooks

- `playbooks/restore_swarm.yml` is used to restore the swarm data
- `playbooks/restore_dtr_meta.yml` is used to restore DTR metadata
- `playbooks/restore_dtr_images.yml` is used to restore DTR images

Backup and restore Docker persistent volumes

There are a number of prerequisites that must be fulfilled before you backup and restore your Docker persistent volumes.

- VSphere clusters should have access to a datastore specifically for backups. This is a separate Virtual Volume created on the HPE 3PAR StoreServ and presented to all the hosts in the vSphere cluster.
- Backup software must be available. HPE Recovery Manager Central and HPE StoreServ is recommended but other customer backup/restore solutions are acceptable.

A number of restrictions also apply:

- Volumes may not be in use when a volume is cloned. Any container that has the volume attached must be paused prior to creating the clone. The container can be resumed once the clone is complete.
- When docker volumes need to be restored from backup, the backup datastore needs to be detached from all vSphere cluster servers prior to restoration.

Persistent storage backup solution

Creating the volume

Docker persistent volumes can be created from a worker node using the following command:

```
docker volume create --driver=vsphere --name=MyVolume@MyDatastore -o size=10gb
```

Cloning the volume

Note

Prior to creating a clone of a volume, any containers accessing the volume should be paused or stopped.

Docker volumes can be cloned to a new datastore:

```
docker volume create --driver=vsphere --name=CloneVolumme@DockerBackup -o clone-
from=MyVolume@MyDatastore -o access=read-only
```

Snapshot and Backup 3PAR Virtual Volumes with HPE Recovery Manager Central and HPE StoreOnce

HPE Recovery Manager Central (RMC) software integrates HPE 3PAR StoreServ All-Flash arrays with HPE StoreOnce Systems to leverage the performance of snapshots with the protection of backups. RMC uses a direct backup model to orchestrate data protection between the array and the backup system without a backup application. When the first full backup is complete, each subsequent backup is incremental, making it significantly faster than traditional backup methods, particularly for higher volumes of data. Backups to HPE StoreOnce are block-level copies of volumes, de-duplicated to save space. Because RMC snapshots are self-contained, fully independent volumes, they can be restored to any 3PAR array in the event of a disaster.

HPE Recovery Manager Central enables you to replicate data from the source storage system (HPE 3PAR StoreServ) to the destination storage system (HPE StoreOnce). The replication is based on point-in-time snapshots.

HPE Recovery Manager Central is installed as a VM on VMware vSphere ESXi. It can be installed on the HPE Synergy platform on a separate (from the Docker Solution) vSphere cluster or external to the Synergy environment as long as the external server has connectivity to the HPE 3PAR StoreServ and HPE StoreOnce. HPE RMC can be installed directly on an ESXi host or can be deployed to a VMware vCenter managed environment. For this solution, the standalone "RMC only" is installed. If HPE RMC is installed in the HPE Synergy environment, iSCSI connection to the HPE 3PAR StoreServ is required.

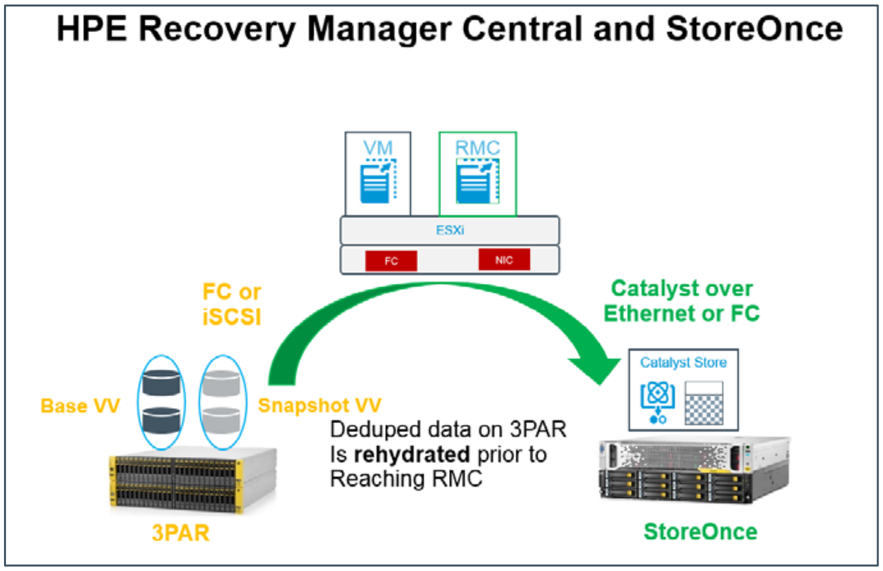

HPE Recovery Manager Central and StoreOnce
VM
RMC
ESXi
FC
NIC
FC or iSCSI
Catalyst over Ethernet or FC
Catalyst Store
Base VV
Snapshot VV
Deduped data on 3PAR Is **rehydrated** prior to Reaching RMC
3PAR
StoreOnce

Figure 10**.** HPE Recovery Manger Central and StoreOnce

- The connectivity between HPE 3PAR StoreServ and HPE RMC for data traffic is over iSCSI.
- The connectivity between HPE StoreOnce and HPE RMC is over CoEthernet (Catalyst OverEthernet)
- The connectivity between HPE RMC, HPE 3PAR StoreServ, and HPE StoreOnce for management traffic is over IP.

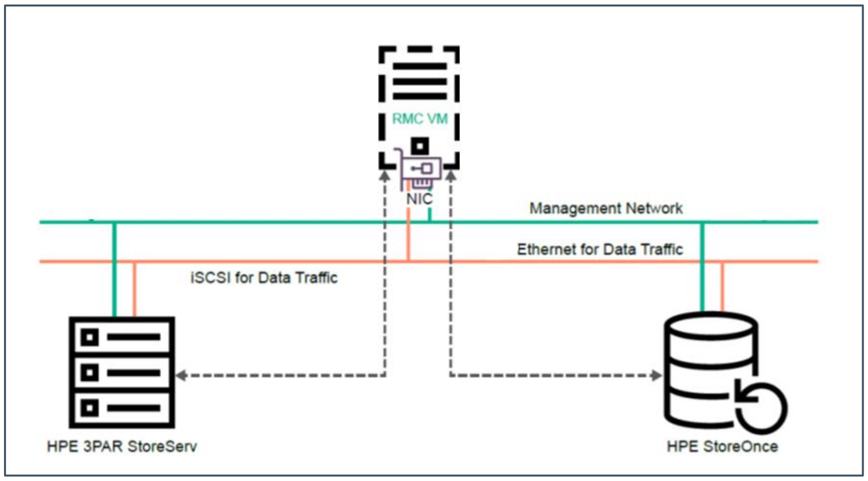
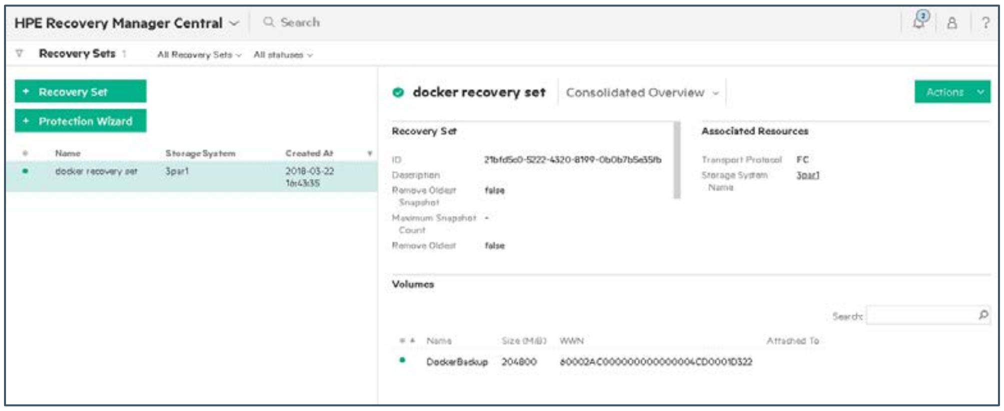


Figure 11**.** Connectivity

Refer to HPE RMC User guide for detailed instructions on setup and configuration of RMC and StoreOnce. When RMC is installed, it can be configured with the Backup Appliance Persona. The Backup persona allows the RMC to manage snapshots and Express Protect Backups. During installation, RMC configuration should specify Data Protection of RMC Core. The initial configuration of backups can be set up using the Protection Wizard. The Protection Wizard assists with creation of a Recovery Set. Create a Recovery Set and select to protect your DockerBackup volume. Once you have created your Recovery Set, the next step is to create Protection Jobs. The Auto Protection Job simplifies the initial configuration of policies. The Auto Protection Job will automatically configure the storage, define default backup policies and protection policies and will schedule snapshots or express protect jobs with the created policies.



Name	Storage System	Created At
docker recovery set	3par1	2018-03-22 16:43:35

Recovery Set

ID	21bfd5c0-5222-4320-8199-0b0b7b5e35fb
Description	
Remove Oldest Snapshot	false
Maximum Snapshot Count	-
Remove Oldest	false

Associated Resources

Transport Protocol	FC
Storage System Name	3par1

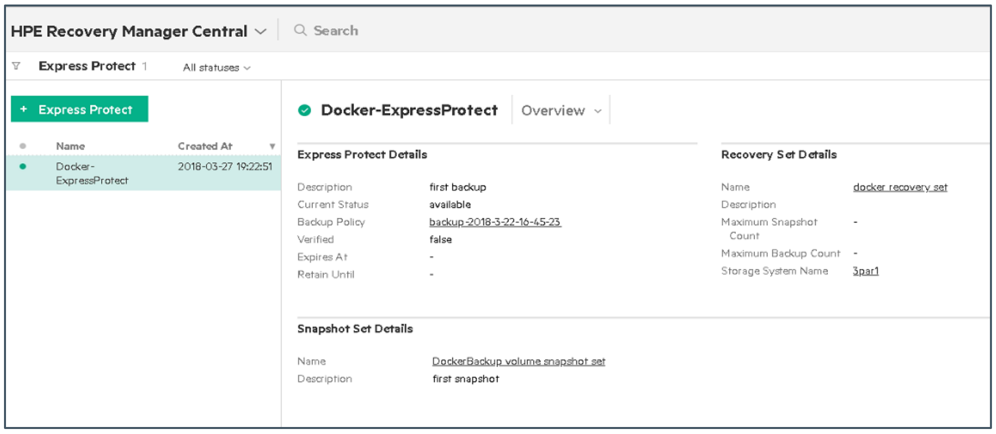
Volumes

Search:

Name	Size (MiB)	WWN	Attached To
DockerBackup	204800	60002AC000000000000004CD0001D322	

Figure 12**.** Recovery Set Overview

RMC uses the Express Protect feature to enable the backup of the snapshot data from the 3PAR array to the StoreOnce system for deduplication and long-term retention.



●	Name	Created At
●	Docker-ExpressProtect	2018-03-27 19:22:51

Express Protect Details

Description	first backup
Current Status	available
Backup Policy	backup-2018-3-22-16-45-23
Verified	false
Expires At	-
Retain Until	-

Recovery Set Details

Name	docker recovery set
Description	
Maximum Snapshot Count	-
Maximum Backup Count	-
Storage System Name	3par1

Snapshot Set Details

Name	DockerBackup volume snapshot set
Description	first snapshot

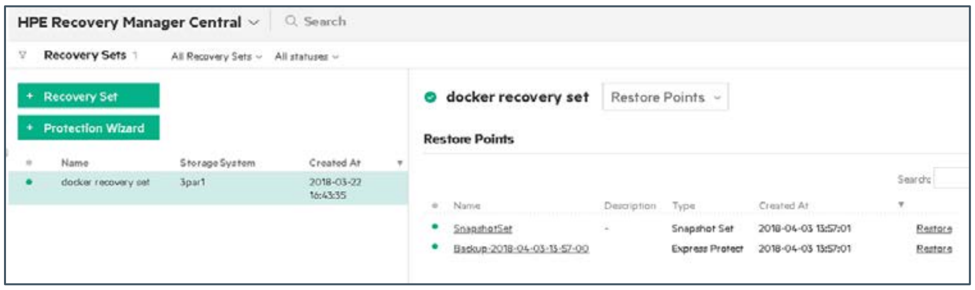
Figure 13. Express Protect

The Express Restore feature restores either snapshots or base volumes.

RMC leverages 3PAR StoreServ SnapDiff technology to create an application-consistent snapshot. Only changed blocks are sent to the StoreOnce system, which minimizes network traffic and saves disk space on the backup system.

Restoring the volume

If a docker persistent storage volume needs to be restored from backup, the 3PAR volume can be restored either from a snapshot saved on the 3PAR or from a backup on StoreOnce. Stop any applications using the docker volume. Use the vSphere Web UI to unmount the datastore from the vSphere cluster. Use RMC to detach the 3PAR virtual volumes prior to restoring the backup. The volume can be restored from a Recovery Set restore point. The Express Protect restore point will restore the volume from the StoreOnce system. A Snapshot Set restore point will restore a 3PAR StoreServ snapshot.



●	Name	Storage System	Created At
●	docker recovery set	3par1	2018-03-22 16:43:35

Restore Points

●	Name	Description	Type	Created At	
●	SnapshotSet	-	Snapshot Set	2018-04-03 13:57:01	Restore
●	Backup-2018-04-03-13-57-00		Express Protect	2018-04-03 13:57:01	Restore

Figure 14. Restore points

Once the 3PAR virtual volume is restored, the volume must be reattached to the vSphere cluster from RMC. After the volume is reattached, the datastore must be mounted. Applications can then access the restored docker volume.

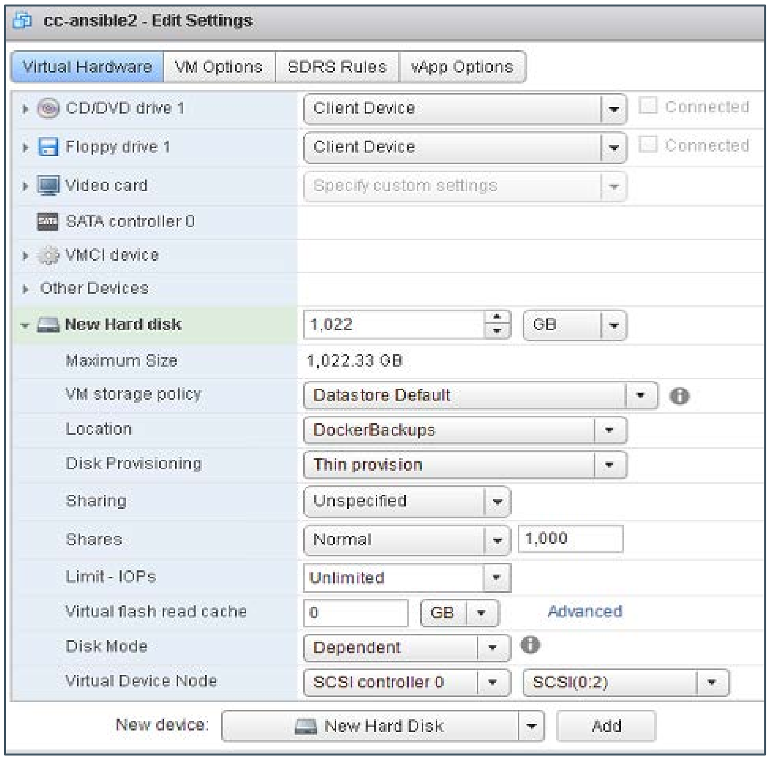
Integrate UCP and DTR backup with HPE RMC and HPE StoreOnce

You can take advantage of HPE Recovery Manager Central and HPE StoreOnce to provide scheduled snapshots and backup protection for the data generated by the backup procedure for Docker UCP and DTR.

Create a datastore from the Backup virtual volume you created and present it to all hosts in the vSphere cluster. This backup datastore is used for storing copies of Docker persistent volumes as well as backups of DTR and UCP.

The Ansible server is used to create backup and restore files for DTR and UCP on the local hard drive. The backup files should be copied to the `DockerBackup` datastore which can be automatically configured for snapshots and offsite backup.

Edit the Ansible server configuration from vCenter. Add a new hard disk and specify the location as the Docker Backup datastore as shown in Figure 19.


cc-ansible2 - Edit Settings

Virtual Hardware | VM Options | SDRS Rules | vApp Options

CD/DVD drive 1	Client Device	Connected
Floppy drive 1	Client Device	Connected
Video card	Specify custom settings	
SATA controller 0		
VMCI device		
Other Devices		
New Hard disk	1,022 GB	
Maximum Size	1,022.33 GB	
VM storage policy	Datastore Default	
Location	DockerBackups	
Disk Provisioning	Thin provision	
Sharing	Unspecified	
Shares	Normal	1,000
Limit - IOPs	Unlimited	
Virtual flash read cache	0 GB	Advanced
Disk Mode	Dependent	
Virtual Device Node	SCSI controller 0	SCSI(0:2)

New device: New Hard Disk | Add

Figure 15**.** Add new hard disk

After the hard disk is added, it is visible from the Linux operating system. From the Ansible server:

```
# ls /dev/sd*
```

The newly added storage should appear as `/dev/sdb`. Now, make a filesystem, ignoring any warnings:

```
# mkfs -t ext4 /dev/sdb
```

Create a mount point for the new disk:

```
# mkdir /dockerbackup
```

Edit the `/etc/fstab` file and add the following line:

```
/dev/sdb /dockerbackup ext4 defaults 0  0
```

After saving the change, mount the new volume using:

```
#mount -a
```

Each time you backup Docker UCP and DTR using the `backup.sh` script, you should copy the generated files from the `/root/backup` folder to `/dockerbackup`. You may wish to add a command to the backup script to automate this process.

The virtual volume used to host the `DockerBackup` datastore can be scheduled for snapshot and backup protection with HPE Recovery Manager Central and HPE StoreOnce as described in the section Backup and restore Docker persistent volumes. Data backed up to HPE StoreOnce can be restored to the HPE 3PAR StoreServ and attached to the Ansible host for recovery.

Summary

TODO

Appendix A: Bill of materials

TODO

Appendix B: Using customer supplied certificates for UCP and DTR

Table 25 list the variables used when configuring customer supplied certificates for UCP and DTR.

Table 25. Customer certs variables

Variable	File	Description
ucp_certs_dir	group_vars/vars	If **ucp_certs_dir** is not defined, UCP is installed with self-signed certificates and DTR is installed with the `--ucp-insecure-tls` switch If **ucp_certs_dir** is defined, this is a folder on the Ansible machine that must contain 3 files: `ca.pem`, this is the root CA certificate in PEM format `cert.pem`, this is the server certificate optionally followed by intermediate CAs `key.pem`, this is the private key that comes with the `cert.pem` certificates
dtr_certs_dir	group_vars/vars	If **dtr_certs_dir** is not defined, DTR is installed with self-signed certificates If **dtr_certs_dir** is defined, this is a folder on the Ansible machine that must contain 3 files: `ca.pem`, this is the root CA certificate in PEM format `cert.pem`, this is the server certificate optionally followed by intermediate CAs `key.pem`, this is the private key that comes with the `cert.pem` certificates

Note

The installation will fail if the `ca.pem`, `cert.pem` and `key.pem` files cannot be found in the folders designated by `dtr_certs_dir` and `ucp_certs_dir` or if they don't constitute valid certificates.

The certificates should specify the names of the FQDNs of the load balancer and the FQDNs of the VMs themselves. This applies to both the UCP server certificate and the DTR server certificate.

Generating and testing certificates

In the example described here we have a root CA named `Example root CA` and an intermediate CA named `Intermediate CA valid 3 years`. The intermediate CA signs the server certificates for UCP and DTR.

Below is the start of the output displayed by running the `openssl x509` utility against the `ca.pem` file (the root CA certificate).

```
[root@ansible ucp_certs]# openssl x509 -text -noout -in ca.pem|head -14
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            0d:07:ca:ea:00:37:77:6e:25:e0:18:3e:0e:db:80:0f:11:cb:1b:3f
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN=Example Root CA
        Validity
            Not Before: Apr 24 20:12:01 2018 GMT
            Not After : Apr 21 20:12:30 2028 GMT
        Subject: CN=Example Root CA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
```

Here is an excerpt from the example `ca.pem` file:

```
-----BEGIN CERTIFICATE-----
MIIFJTCCAw2gAwIBAgIUDQfK6gA3d24l4Bg+DtuADxHLGz8wDQYJKoZIhvcNAQEL
BQAwGjEYMBYGA1UEAxMPRXhhbXBsZSBSb290IENBMB4XDTE4MDQyNDIwMTIwMVoX
...
...
uXzYbCtU6Jt9B3fayAeWWswQv+jQSzuuA3reOM1x838iIZWDx93f4yLJWLJl7xsY
btvKBmqKDCsAqsQLFLnNj/JyYq4e9a6Xxcyn9FXNpzuEsfjfNGHn+csY+w3f987T
MNviy376xZbyAc1CV5kgmnZzjU5bDkgT8Q==
-----END CERTIFICATE-----
```

The `cert.pem` file should contain the server certificate itself, followed by your intermediate CA's certificate. The following example shows how to extract the intermediate CA certificate from `cert.pem` and to save it to a file named `intca.pem`. Using the `openssl x509` utility, you can display the content of the `intca.pem` file in human readable form. This certificate was signed by the example CA above (`Issuer = 'Example Root CA'`).

```
[root@ansible ucp_certs]# openssl x509 -text -noout -in intca.pem|head -14
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            6b:1e:0c:86:20:cf:f0:88:d2:52:0d:5d:b9:56:fa:91:87:a0:49:18
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN=Example Root CA
        Validity
            Not Before: Apr 24 20:12:09 2018 GMT
            Not After : Apr 23 20:12:39 2021 GMT
        Subject: CN=Intermediate CA valid 3 years
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
```

Here is an excerpt from the `incta.pem` file showing the example Intermediate CA certificate:

```
-----BEGIN CERTIFICATE-----
MIIFcjCCA1qgAwIBAgIUax4MhiDP8IjSUg1duVb6kYegSRgwDQYJKoZIhvcNAQEL
BQAwGjEYMBYGA1UEAxMPRXhhbXBsZSBSb290IENBMB4XDTE4MDQyNDIwMTIwOVoX
...
...
o2tL5nwR7ROiAr/kk9MIRzWrLNbc4cYth7jEjspU9dBqsXgsTozzWlwqI9ybZwvL
Ni1JnZandVlyQdoOaB2M/1DNFfKvwW3JeArKvDA9j95n/BWFTjoZ+YOz9pYit6T7
1GCGu3be
-----END CERTIFICATE-----
```

The `openssl x509` utility will only decrypt the first certificate found in `cert.pem`, so you don't need to extract the server certificate from `cert.pem`. In this example, the server certificate is signed by the intermediate CA above. Note the `Subject Alternate Names: hpe-ucp.cloudra.local` is the FQDN of the UCP load balancer, and the other names are those of the UCP instances (`hpe-ucp01.cloudra.local`, `hpe-ucp02.clodura.local`, `hpe-ucp03.cloudra.local`).

```
[root@ansible ucp_certs]# openssl x509 -text -noout -in server.pem
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            25:d9:f8:1d:9b:1d:23:f1:21:56:54:f2:43:cc:4f:0e:73:22:be:ec
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN=Intermediate CA valid 3 years
        Validity
            Not Before: Apr 24 20:17:30 2018 GMT
            Not After : Apr 24 20:18:00 2019 GMT
        Subject: O=HPE, OU=CloudRA Team, CN=hpe-ucp.cloudra.local
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                CA Issuers - URI:http://localhost:8200/v1/intca
      ( portions removed )

            X509v3 Subject Alternative Name:
                DNS:hpe-ucp.cloudra.local, DNS:hpe-ucp01.cloudra.local, DNS:hpe-
ucp02.cloudra.local, DNS:hpe-ucp03.cloudra.local
```

The following excerpts from `cert.pem` show the first certificate which is the server certificate itself and the second certificate which is the intermediate CA's certificate.

```
-----BEGIN CERTIFICATE-----
MIIFGTCCAwGgAwIBAgIUJdn4HZsdI/EhVlTyQ8xPDnMivuwwDQYJKoZIhvcNAQEL
BQAwKDEmMCQGA1UEAxMdSW50ZXJtZWRpYXRlIENBIHZhbGlkIDMgeWVhcnMwHhcN
...
...
sOR4I3Qnc5OoNISng5l7wW1d4RMMwmXQhG1H5QKAUjHfJXH4bNtIzKxw/zGTVr4Z
llYKbEwJcgAvvfkn+w==
-----END CERTIFICATE-----
```

```
-----BEGIN CERTIFICATE-----
MIIFcjCCA1qgAwIBAgIUax4MhiDP8IjSUg1duVb6kYegSRgwDQYJKoZIhvcNAQEL
BQAwGjEYMBYGA1UEAxMPRXhhbXBsZSBSb290IENBMB4XDTE4MDQyNDIwMTIwOVoX
...
...
Ni1JnZandVlyQdoOaB2M/1DNFfKvwW3JeArKvDA9j95n/BWFTjoZ+YOz9pYit6T7
1GCGu3be
-----END CERTIFICATE-----
```

Finally, here is an excerpt from key.pem, the private key that goes with the server certificate.

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEA5rmmb52ufE8Oa3cXhY2HSRZNazb7/fipXY1rZ+U5+rJv9BN5
d/X3NTroSE8/PvoS/maGkHCnURGNqbu/G2umKN/tm/eSpDY861YnGWxj+bcOgtiU
...
...
AOSGidSMk3hFX1Iaftgx4EUGbrzZO7I8M5RO64U1aMFNFyj4XghJ2mZTdNelwNBw
pr/fYulyi5lYPalQHYH3OyvNqQQ3arEbTbZp8hEyYOgxtZRXmmaoqOY=
-----END RSA PRIVATE KEY-----
```

Verify your certificates

The playbooks do not verify the validity of the certificate files you supply so you should verify them manually before you start your deployment.

Verify that the private and the server cert match

On the Ansible box, run the following commands:

```
ckcert=$(openssl x509 -noout -modulus -in cert.pem | openssl md5)
ckkey=$(openssl rsa -noout -modulus -in key.pem| openssl md5)
if [ "$ckkey" == "$ckcert" ] ; then echo "Private key and Certificate match" ; else echo
 "STOP! Private Key and Certificate don't match" ; fi
```

Verify that the server certificate was signed by the CA

Extract all but the first certificate from `cert.pem` (i.e. extract the certs for the intermediate CA authorities) into the file `int.pem`

```
sed -e '1,/-----END CERTIFICATE-----/d' cert.pem >intca.pem
```

Combine `intca.pem` and `ca.pem` to form `cachain.pem`:

```
cat intca.pem ca.pem > cachain.pem
```

Finally, verify that `cert.pem` was signed by the CA or by an intermediate CA:

```
openssl verify -verbose -CAfile cachain.pem  cert.pem
```

A successful check will generate output similar to:

```
[root@ansible ucp_certs]# cat intca.pem ca.pem > cachain.pem
[root@ansible ucp_certs]# openssl verify -verbose -CAfile cachain.pem  cert.pem
cert.pem: OK
```

An unsuccessful check will generate output similar to:

```
[root@ansible ucp_certs]# openssl verify -verbose -
CAfile cachain.pem  certsignedbyanotherca.pem
certsignedbyanotherca.pem: O = HPE, OU = CloudRA Team, CN = hpe-ucp.cloudra.local
error 20 at 0 depth lookup:unable to get local issuer certificate
```

Appendix C: Enabling SSL between the universal forwarders and the Splunk indexers using your certificates

Overview of enabling SSL

The procedure for enabling SSL between the universal forwarders and the Splunk indexers using your certificates is described below. In summary, the following steps are required:

1. Set the variable `splunk_ssl` to `yes` in `group_vars/vars`
2. Put your root CA certificate and your server certificate files in `/root/Docker-Synergy/ops/files/splunk/linux/SPLUNK_HOME/etc/mycerts`
3. Uncomment the `[sslConfig]` stanza in the file `/files/splunk/linux/SPLUNK_HOME/etc/system/local/server.conf`

Limitations

SSL only works with Linux worker nodes. The Universal Forwarders verify that the indexers they connect to have a certificate signed by the configured root CA and that the Common Name in the certificate presented by the indexer matches its FQDN as listed by the variable `splunk_architecture_forward_servers`.

Prerequisites

Configure your indexers to use SSL on port 9998. The following is an example `inputs.conf` file located in `$SPLUNK_HOME/etc/system/local` that enables SSL on port 9998 and configures the certificate file for use by the indexer itself, in this instance `/opt/splunk/etc/mycerts/indexer.pem`.

```
[splunktcp-ssl://9998]
disabled=0
connection_host = ip

[SSL]
serverCert=/opt/splunk/etc/mycerts/indexer.pem
#requireClientCert = true
#sslAltNameToCheck = forwarder,forwarder.cloudra.local

[tcp://1514]
connection_host = dns
sourcetype = ucp
```

For more information, see the documentation at https://docs.splunk.com/Documentation/Splunk/7.0.2/Security/ConfigureSplunkforwardingtousesignedcertificates. In addition, you can see how to create your own certificates and the content of the file designated with `serverCert` at http://docs.splunk.com/Documentation/Splunk/7.0.2/Security/Howtoself-signcertificates.

In this instance, the folder `mycerts` was created under `/opt/splunk/etc` and the file `indexer.pem` was copied to this folder.

Indexers are configured with the Root CA cert used to sign all certificates. This can be achieved by editing the file `server.conf` in `$SPLUNK_HOME/etc/system/local` on your indexer(s). The following code block shows the relevant portion of this file where `sssRootCaPath` is pointing to the root CA certificate.

```
[sslConfig]
sslRootCAPath = /opt/splunk/etc/mycerts/ca.pem
```

Note

In order to be able to download and install additional applications, you may want to append the file `$SPLUNK_HOME/auth/appsCA.pem` to your `ca.pem` file. If you don't do this, the Splunk UI will make this suggestion when you attempt to `Find more apps`.

Splunk should be restarted on the indexers if you had to make these changes (see the Splunk documentation for more information).

Before you deploy

Generate the forwarder certificate and name it `forwarder.pem`. Make sure that you copy the root CA certificate to `ca.pem`

Copy both the `ca.pem` and the `forwarder.pem` files to `files/splunk/linux/SPLUNK_HOME/etc/mycerts/` (overwriting any existing files).

Edit the file `server.conf` in the folder `files/splunk/linux/SPLUNK_HOME/etc/system/local` and uncomment the last two lines as suggested in the file itself. Your file should look like this:

```
#
# uncomment the section below if you want to enable SSL
#
```

```
[sslConfig]
sslRootCAPath = /opt/splunkforwarder/etc/mycerts/ca.pem
```

Set splunk_ssl to yes in the file group_vars/vars, uncommenting the line if required. Make sure that the splunk_architecture_forward_servers list specifies all your indexers together with the port that was configured to accept SSL:

```
monitoring_stack: splunk
splunk_ssl: yes
splunk_architecture_forward_servers:
- indexer1.cloudra.local:9998
- indexer2.cloudra.local:9998
```

Hybrid environment Linux/ Windows

Currently, you cannot deploy your own certificates for use by the Universal Forwarders deployed on Windows machines. If you want to have your Linux machines in a hybrid deployment to use SSL, proceed as follows.

Comment out the `splunk_architecture_forward_servers` variable (and its values) from `group_vars/vars`

```
monitoring_stack: splunk
splunk_ssl: yes
#splunk_architecture_forward_servers:
#  - hpe2-ansible.cloudra.local:9998
```

Create a file named `vms.yml` in the folder `group_vars` and specify the list of forward servers to use by the Linux servers. This list is typically the same as the one used for Windows servers but specifies a TCP port that enables SSL.

```
splunk_architecture_forward_servers:
- hpe2-ansible.cloudra.local:9998
```

Edit the `group_vars/win_worker.yml` file and specify the list of forward servers to be used by the Windows servers. This list is typically the same as the one used for Linux servers but specifies a TCP port that does not enable SSL.

```
splunk_architecture_forward_servers:
- hpe2-ansible.cloudra.local:9997
```

Appendix D: How to check that certs were deployed correctly

The following commands should return the CA certificates used by UCP/DTR. This certificates is the same as the one pointed to by the `--cacert` switch. If the deployment was not successful, `curl` will output something like **Output 2.**

```
# curl --cacert <ucp_certs_dir>/ca.pem https://<your ucp fqdn>/ca
# curl --cacert <dtr_certs_dir>/ca.pem https://<your dtr fqdn>/ca
```

Output 1: certificates successfully deployed (content will depend on your own CA certificate)

```
-----BEGIN CERTIFICATE-----
MIIDyTCCArGgAwIBAgIUUeo+H6xGSB7/9gqq9T2SUwJPLggwDQYJKoZIhvcNAQEL
BQAwbDELMAkGA1UEBhMCRlIxFTATBgNVBAcTDFRoZSBJbnRlcm5ldDETMBEGA1UE
ChMKQ2hyaXN0b3BoZTEUMBIGA1UECxMLQ0EgU2VydmljZXMxGzAZBgNVBAMTEkNo
...
XkJ8WcsHocJO8J9J3RaWsM2BQc7wRntJcOkA7ooTH13OtQTP1jFcQp5xNdI4J3Mz
j9BAYERjkGqu7v9tfOem99oVGUal2Opu4r73eWUm1mL948xuw6PgiRSLZrXhn/RS
uvFVnS/vPYJozOXIZA==
-----END CERTIFICATE-----
```

Output 2: certificates were not successfully deployed

```
curl: (60) Peer's Certificate issuer is not recognized.
More details here: http://curl.haxx.se/docs/sslcerts.html
...
```

Enable certs for browser (Windows 2016 example)

Choose `Manage computer certificates` in the control panel

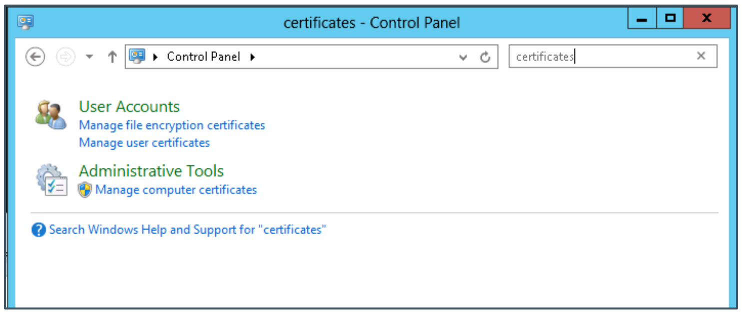

certificates - Control Panel
Control Panel
certificates
User Accounts
Manage file encryption certificates
Manage user certificates
Administrative Tools
Manage computer certificates
Search Windows Help and Support for "certificates"

Figure 16. Manage computer certificates

Import the `ca.pem` for UCP into the Trusted Root Certification Authorities

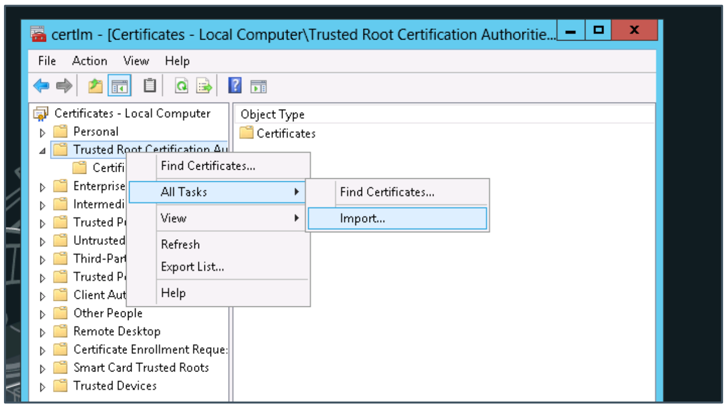
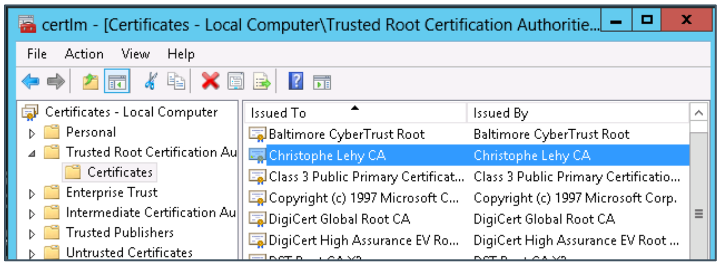


Figure 17. Import the ca.pem

It should show up in the list of certificates:


certlm - [Certificates - Local Computer\Trusted Root Certification Authoritie...
File
Action
View
Help
Certificates - Local Computer
Personal
Trusted Root Certification Au
Certificates
Enterprise Trust
Intermediate Certification Au
Trusted Publishers
Untrusted Certificates

Issued To	Issued By
Baltimore CyberTrust Root	Baltimore CyberTrust Root
Christophe Lehy CA	Christophe Lehy CA
Class 3 Public Primary Certificat...	Class 3 Public Primary Certificatio...
Copyright (c) 1997 Microsoft C...	Copyright (c) 1997 Microsoft Corp.
DigiCert Global Root CA	DigiCert Global Root CA
DigiCert High Assurance EV Ro...	DigiCert High Assurance EV Root ...

Figure 18. List certificates

You may need to restart your browser for this to take effect:

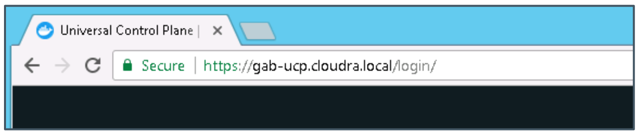

Universal Control Plane |
Secure
https://gab-ucp.cloudra.local/login/

Figure 19**.** Secure HTTPS