# Table of Contents

Lab Overview - HOL-1830-02-CNA - vSphere Integrated Containers - Getting Started ..... 2

Lab Guidance .................................................................................................................. 3

Module 1: Introduction to vSphere Integrated Containers (15 minutes) ......................... 9

  Introduction.................................................................................................................. 10
  Tradeoffs between Containers and Virtual Machines................................................. 11
  The Best of Both Worlds......................................................................................... 15
  Looking Under the Hood......................................................................................... 18
  Conclusion............................................................................................................... 23

Module 2: Setting up vSphere Integrated Containers (15 minutes) ................................. 24

  Introduction................................................................................................................ 25
  Overview of Components......................................................................................... 26
  Make Sure All Components are Running .............................................................. 28
  Installation and Configuration................................................................................. 30
  Conclusion............................................................................................................... 35

Module 3: Using vSphere Integrated Containers (15 minutes)......................................... 36

  Make Sure All Components are Running .............................................................. 37
  Deploying a simple container............................................................................. 39
  Deploying an Application....................................................................................... 48
  Conclusion............................................................................................................... 61

Module 4: VMware Harbor - An Enterprise-class Registry Server (15 minutes) ............. 62

  Introduction to Registry Servers and VMware Harbor ......................................... 63
  Make Sure All Components are Running .............................................................. 68
  A Quick Tour of the Harbor UI.............................................................................. 70
  Projects, users and roles....................................................................................... 78
  Conclusion............................................................................................................... 89

Module 5: Container Management with VMware Admiral (15 minutes)......................... 90

  An Introduction to VMware Admiral...................................................................... 91
  Make Sure All Components are Running .............................................................. 95
  A Quick Tour Through the UI.............................................................................. 97
  Conclusion............................................................................................................... 112
Lab Guidance

Note: It will take more than 75 minutes to complete this lab. You should expect to only finish 2-3 of the modules during your time. The modules are independent of each other so you can start at the beginning of any module and proceed from there. You can use the Table of Contents to access any module of your choosing.

The Table of Contents can be accessed in the upper right-hand corner of the Lab Manual.

This lab provides an introduction to vSphere Integrated Containers (VIC). Throughout the lab, we will show how to install the components, configuring them and how to deploy a basic container using this technology. In module 3, we will also look at how to deploy a more realistic application and discuss things like persistent volumes. Finally, we will discuss the container control plane (Admiral) and VMware's Enterprise Registry solution (Harbor).

Lab Module List:

- **Module 1: Introduction to vSphere Integrated Containers (15 minutes)**
- **Module 2: Setting up vSphere Integrated Containers (15 minutes)**
- **Module 3: Using vSphere Integrated Containers (15 minutes)**
- **Module 4: Introduction to Harbor (15 minutes)**
- **Module 5: Introduction to Admiral (15 minutes)**

Lab Captain:

- **Pontus Rydin, Principal System Engineer, USA**

This lab manual can be downloaded from the Hands-on Labs Document site found here:

[http://docs.hol.vmware.com](http://docs.hol.vmware.com)

This lab may be available in other languages. To set your language preference and have a localized manual deployed with your lab, you may utilize this document to help guide you through the process:

Location of the Main Console

1. The area in the RED box contains the Main Console. The Lab Manual is on the tab to the Right of the Main Console.
2. A particular lab may have additional consoles found on separate tabs in the upper left. You will be directed to open another specific console if needed.
3. Your lab starts with 90 minutes on the timer. The lab can not be saved. All your work must be done during the lab session. But you can click the EXTEND to increase your time. If you are at a VMware event, you can extend your lab time twice, for up to 30 minutes. Each click gives you an additional 15 minutes. Outside of VMware events, you can extend your lab time up to 9 hours and 30 minutes. Each click gives you an additional hour.

Alternate Methods of Keyboard Data Entry

During this module, you will input text into the Main Console. Besides directly typing it in, there are two very helpful methods of entering data which make it easier to enter complex data.
Click and Drag Lab Manual Content Into Console Active Window

You can also click and drag text and Command Line Interface (CLI) commands directly from the Lab Manual into the active window in the Main Console.

Accessing the Online International Keyboard

You can also use the Online International Keyboard found in the Main Console.

1. Click on the Keyboard Icon found on the Windows Quick Launch Task Bar.
Click once in active console window

In this example, you will use the Online Keyboard to enter the "@" sign used in email addresses. The "@" sign is Shift-2 on US keyboard layouts.

1. Click once in the active console window.
2. Click on the Shift key.

Click on the @ key

1. Click on the "@ key".

Notice the @ sign entered in the active console window.
Activation Prompt or Watermark

When you first start your lab, you may notice a watermark on the desktop indicating that Windows is not activated.

One of the major benefits of virtualization is that virtual machines can be moved and run on any platform. The Hands-on Labs utilizes this benefit and we are able to run the labs out of multiple datacenters. However, these datacenters may not have identical processors, which triggers a Microsoft activation check through the Internet.

Rest assured, VMware and the Hands-on Labs are in full compliance with Microsoft licensing requirements. The lab that you are using is a self-contained pod and does not have full access to the Internet, which is required for Windows to verify the activation. Without full access to the Internet, this automated process fails and you see this watermark.

This cosmetic issue has no effect on your lab.

Look at the lower right portion of the screen
Please check to see that your lab is finished all the startup routines and is ready for you to start. If you see anything other than "Ready", please wait a few minutes. If after 5 minutes your lab has not changed to "Ready", please ask for assistance.
Module 1: Introduction to vSphere Integrated Containers (15 minutes)
Introduction

In this module, we will discuss the benefits and shortcomings of containers as well as taking a brief look at the architecture of vSphere Integrated Containers.

Notice: This entire module is all reading and no interactive exercises. If you feel you have a good grasp on the concepts behind vSphere Integrated Containers, you're welcome to proceed directly to module 2. However, since vSphere Integrated Containers, Harbor and Admiral are all fairly new products, we believe that most people would benefit from reading through this module.
Tradeoffs between Containers and Virtual Machines

Contrary to popular belief, containers are not a drop-in replacement for virtual machines. In this lesson, we will discuss the various benefits and drawbacks of containers vs. virtual machines and outline the use cases where each technology excels.

A quick review of the differences between virtual machines and containers

Let's quickly review how virtual machines and containers differ. As you can see from the diagram above they are somewhat similar but differ in some very important areas.

Virtual Machines on a Bare Metal Hypervisor (Such as vSphere)

A component called the hypervisor provides a software equivalent of the server hardware that's presented to the guest operating systems. The hypervisor provides a perfect emulation of the compute, storage and networking hardware with a very low overhead (typically 1-5%). Guest operating systems run unmodified on top of the virtualized hardware. In fact, the guest Operating System is unaware of the fact that it's running on top of virtualized hardware. The same goes for any libraries running on top of the Operating System.
Since the hardware is fully virtualized, we can introduce features like virtualized storage, virtualized networking, intelligent memory management and workload mobility without affecting the guest OS or the application. And since hypervisors and hardware virtualization are very mature technologies, robust and proven technologies such as vSAN, NSX and vMotion have become the industry standard.

Isolation between workloads is carried out by the hypervisor at the virtual hardware level. Since any given OS can only see its own slice of the virtualized hardware, the isolation between workloads is perfect as long as the hypervisor doesn't leak data between virtual machines. Although hypervisors are complex pieces of software, the footprint is relatively small and leaks between virtual machines are extremely rare.

**Containers (such as Docker)**

Containers do not run on virtualized hardware and do not need a hypervisor. Instead, the utilize two mechanisms in the Linux kernel known as cgroups and namespaces. Simply put, cgroups is a mechanism for dividing and sharing hardware resources and namespaces is a way to provide some level of isolation. In essence, namespaces allow you to declare a sandbox within the operating system where you don't see the resources of other namespaces. This type of OS-level virtualization has been around for a long time and you may have encountered it as Solaris Zones or IBM LPARs in earlier generations of computing.

Each container runs an operating system "flavor". It's important to note that this is not a full OS as in the virtual machine case, but a set of libraries and commands that give application the impression of running inside a specific OS.

An important difference from virtual machines is the level of isolation. Loosely speaking, a container is simply a process running in the host OS and isolation is done at the OS kernel level. In other words, each kernel function has to explicitly check which namespace it executes in and take measures to isolate it from others. Because of this, the number of points where a leak between workloads can happen is considerably larger than with a hypervisor. Also, isolation is dependent on the host OS and you have to take great care to run only host Oses that are known to be safe and that have all current patches installed. Although leaks between containers are relatively rare, they have happened and given rise to the saying "Containers contain. Until they don't".
Pros and cons

<table>
<thead>
<tr>
<th></th>
<th>Virtual Machines</th>
<th>Containers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isolation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Agility</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Image sharing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Infrastructure virtualization</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Developer friendliness</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Let’s explore some of the key advantages and disadvantages of virtual machines and containers.

**Isolation**

- Virtual machines - Isolated at the virtual hardware level. A single, smaller code base with fewer points of possible leakage (+)
- Containers - Isolated at the OS kernel level. Larger codebase, larger surface for leaks and harder to keep updated (-)

**Security**

- Virtual machines - Strong isolation, network microsegmentation, virtual firewalls, storage encryption, hardware rooted trust (+)
- Containers - Weaker isolation, lacks mature network microsegmentation and virtual firewalls. No native storage encryption or hardware rooted trust (-)

**Performance**

- Virtual machines - Slight performance penalty due to hardware virtualization, but mostly cancelled out by better scheduling and memory management by the hypervisor (=)
- Containers - Virtually no performance penalty, but lacks some of the optimizations offered by a hypervisor (=)

**Management**

- Virtual machines - A very mature management eco-system exists from multiple vendors (+)
- Containers - Management tools are still emerging and very immature (-)

**Agility**
• Virtual machines - Can take several minutes to spin up. Images are typically 1GB-100GB in size. (-)
• Containers - Take milliseconds to seconds to spin up. Images are typically 1MB-100MB in size(+) 

Image sharing
• Virtual machines - Requires a large amount of storage due to image size. No standard registry exists. (-)
• Containers - Modest storage requirements. Robust public and private registries exist (+)

Infrastructure virtualization
• Virtual Machines - Very mature technologies, such as NSX and vSAN exist to virtualize all infrastructure. (+)
• Containers - Infrastructure virtualization is still emerging (-)

Developer friendliness
• Virtual machines - Somewhat slow to set up and more resource heavy. May require approval to create (-)
• Containers - Extremely quick to create and lean on resources. Can typically be created without approval (+)
The Best of Both Worlds

In this lesson, we will explore how vSphere Integrated Containers offers the a way of combining the benefits of virtual machines with those of containers.

Introducing vSphere Integrated Containers

As we learned in the previous lesson, both virtual machines and containers have benefits and drawbacks. Furthermore, the drawbacks aren't just at a technical level. Most organizations have made substantial investments in building expertise around virtual machines. Moving towards a container based infrastructure would require new investments in training and management tools.

But what if we could make a container look just like a virtual machine? We could give developers the agility and speed they need, while allowing operations to reuse the tools, processes and people they've already invested in. This is exactly the solution that vSphere Integrated Containers offers!
vSphere Integrated Containers turn container into objects that look just like container when viewed from a developer’s perspective and looks just like a virtual machine when viewed from an operator’s perspective. Since vSphere Integrated Containers expose the Docker API, all developer tools, scripts and processes will continue to work. And since they behave just like virtual machines, vCenter, NSX, vRealize Operations, vSAN, vMotion and other familiar technologies are suddenly applicable to containers!

**Key Benefits**

**Run Containerized Apps Alongside Existing Workloads**

By leveraging the existing capabilities of vSphere, IT Ops can run containerized apps alongside traditional VMs on the same infrastructure without having to build out a separate, specialized container infrastructure stack.

**Combine Portability with Security, Visibility and Management**

By running containers as VMs, IT teams can leverage vSpheres core capabilities such as enterprise-class security, networking, storage, resource management, and compliance that are essential to running containerized apps in a production environment.

**Leverage Your Existing Infrastructure, Scale Easily**

Avoid costly and time-consuming re-architecture of your infrastructure that results in silos. Scale application deployments instantly.

**Provide Developers with a Docker Compatible Interface**

Developers already familiar with Docker can develop applications in containers, by using a Docker compatible interface and provision them through the self-service management portal or UI.

**Components of vSphere Integrated Containers**

In order to provide a full ecosystem for containers, it is necessary to implement more than just the mechanism for running containers as virtual machines. For example, we need a robust, enterprise-scale container registry as well as a user interface for managing containers and container hosts. Note that although these are technically separate open source projects, they are fully integrated into the commercial distribution of vSphere Integrated Containers and share the same user interface.

**vSphere Integrated Containers Engine**

Enterprise container runtime for vSphere that allows developers who are familiar with Docker to develop in containers and deploy them alongside traditional VM-based
workloads on vSphere clusters. vSphere admins can manage these workloads through the vSphere Web Client in a way that is familiar.

**VMware Harbor**

Enterprise container registry that stores and distributes container images. Harbor extends the Docker Distribution open source project by adding the functionalities usually required by an enterprise, such as security, identity and management.

**VMware Admiral**

Management portal that provides a UI for dev teams to provision and manage containers. Cloud administrators can manage container hosts and apply governance to their usage, including capacity quotas and approval workflows. Advanced capabilities are available when integrated with vRealize Automation.
Looking Under the Hood

In this lesson we will look closer at how the vSphere Integrated Containers engine enables all the benefits of a virtual machine for a container. We will explore the concept of a container VM and the Virtual Container Host (VCH).

Architectural Overview

As we mentioned before, a vSphere Integrated Container looks just like a virtual machine. Here's the secret: A vSphere Integrated Container IS a virtual machine!

Instead of running a container host as a virtual machine and deploying several containers onto that same host, we are deploying a single container per virtual machine.
In order to run a container as a virtual machine, we need to address the overhead a virtual machine introduces. A conventional virtual machine carries its own operating system, complete with all the libraries and user tools. This makes them too heavy for running a single container per virtual machine. So instead of deploying a full-fledged virtual machine, we deploy the bare minimum: A slimmed-down Linux Kernel with just enough code to run a Docker image. The image itself is converted to a standard VMDK and attached to the VM. When the machine is booted, it simply jumps straight into the payload code of the container. Note that there's no Docker engine involved. The code of the container runs right on top of the kernel, along with the libraries for the OS flavor and a minimal set of supporting services. By employing techniques such as Linked Clones, we can get container VMs to start within a few seconds (sometimes as quickly as 1-2 seconds).
The Virtual Container Host is responsible for managing the container VMs and for providing a Docker API endpoint. Specifically, it serves the following purposes.

- **Docker API endpoint** - Exposes the Docker host API. From a Docker client point of view, the Virtual Docker Host looks exactly like a standard Docker host. To use the standard command line tools, simply set DOCKER_HOST to point to your virtual Docker host. Commands like `docker run`, `docker volume` and `docker net` will work exactly as they do with a standard Docker host.

- **Internal orchestration** - When a docker container is started, the Virtual Container Host is responsible for coordinating the creation of a Container VMs, container networks (as NSX networks or port groups) and volumes (as VMDKs on datastores). Resources are drawn from a resource pool that's tied to the Virtual Container host.

- **Port mapping and routing** - When a container port is mapped to a host port, the Virtual Docker Host is responsible for listening on that port and routing traffic to the corresponding container VM.

Since all the orchestration and routing is hidden behind a standard Docker API, clients will continue to work exactly as they do with a standard Docker host.
Enterprise Registry (Harbor)

A Docker Registry is responsible for storing and indexing any images used by containers. Registries can be public or private. When you type e.g. `docker run busybox`, a public registry (hosted by Docker) is queried for an image by that name and the binary data making up the image is downloaded if it's not already cached on your machine. This works fine as long as we are using public domain/open source images, but would not work for proprietary images. In those cases you would have to host your own registry.

While vSphere Integrated Containers can interoperate with any Docker Registry, organizations using containers in production typically need a hardened Enterprise class Registry.

The Harbor Enterprise Registry offers the following features:

- **Role based access control**: Users and repositories are organized via 'projects' and a user can have different permission for images under a project.
- **Policy based image replication**: Images can be replicated (synchronized) between multiple registry instances. Great for load balancing, high availability, multi-datacenter, hybrid and multi-cloud scenarios.
- **LDAP/AD support**: Harbor integrates with existing enterprise LDAP/AD for user authentication and management.
- **Image deletion & garbage collection**: Images can be deleted and their space can be recycled.
• **Notary**: Image authenticity can be ensured.
• **Graphical user portal**: User can easily browse, search repositories and manage projects.
• **Auditing**: All the operations to the repositories are tracked.
• **RESTful API**: RESTful APIs for most administrative operations, easy to integrate with external systems.
• **Easy deployment**: Provide both an online and offline installer.

**Management Framework (Admiral)**

![Diagram of Management Framework (Admiral)]
Conclusion

In this module, we have discussed the following:

- Tradeoffs between containers and virtual machines
- How to get the best of both worlds with vSphere Integrated Containers
- An overview of the vSphere Integrated Containers Architecture

You’ve finished Module 1

Congratulations on completing Module 1.

Proceed to any module below which interests you most.

- **Module 2: Setting up vSphere Integrated Containers (15 minutes)**
- **Module 3: Using vSphere Integrated Containers (15 minutes)**
- **Module 4: Introduction to Harbor (15 minutes)**
- **Module 5: Introduction to Admiral (15 minutes)**

How to End Lab

To end your lab click on the **END** button.
Module 2: Setting up vSphere Integrated Containers (15 minutes)
Introduction

In this module, we are discussing the components of vSphere Integrated Containers. We will conclude it with an interactive tour of the installation and configuration.
Overview of Components

The goal of this lesson is to provide a basic understanding of the main components of vSphere Integrated Containers. In this lesson, we will cover the following:

- The vSphere Integrated Container appliance
- Image and volume stores
- Bridge networks and public networks
- Virtual Container hosts and their relationship to resource pools

What gets installed?

To bootstrap vSphere Integrated Containers, we install a single OVA that creates a virtual appliance. This appliance has three main purposes:

- Providing an image registry
- Providing a management interface
- Providing a repository of appliances and plugins for the rest of the installation.

Once the appliance is running, we can use it to install the remaining components. The core of the vSphere Integrated Containers is the Virtual Container Host. This component is responsible for orchestrating the operations needed to run containers, as well as for exposing the Docker API to clients. Your environment may contain more than one Virtual
Container Host. Each Virtual Container Host is associated with a vSphere Resource Pool for fine-grained control of resource allocations.

In addition to this, a few additional vSphere objects are needed:

- A datastore designated as Image Store. This is where the Virtual Container Hosts store locally cached Docker images.
- A datastore designated as Volume Store. This is where the Virtual Container Hosts store the volumes of containers.
- A bridge network for communication between containers
- A public network for communication between containers and other applications/clients.

In the following sections, we will demonstrate how to install and configure each of these components.
Make Sure All Components are Running

Due to timing issues in the lab affecting the startup order of components, it is advisable to restart the containers that run the local Docker registry. You only need to do this once per lab, so if you have already done this as part of another module, you may skip this step. Should you experience problems with the registry refusing to authenticate you or timing out, you can always issue this command again.

Log in to Docker CLI

1. On the desktop, double-click the Docker CLI icon.
2. Log in with root and press the Enter key. You should be automatically logged.
Make Sure All Components are Running

1. At the command prompt, enter the following and press enter:

   ```bash
   ssh -o "StrictHostKeyChecking no" root@192.168.100.102 "systemctl restart docker"
   ```

2. When prompted for a password, use **VMware1!**

   It will take a minute or two, but you should be brought back to the command prompt.
Installation and Configuration

In this lesson we will learn how to install the vSphere Integrated Containers components from an OVA file and how to do the basic setup. Specifically, we will learn how to:

- Install the OVA
- Create a virtual container host
- Verify the installation

Installing the OVA

Notice: The OVA is already installed in your lab. The steps below are for illustration only and should not be carried out in the lab!

You install vSphere Integrated Containers by deploying an OVA appliance. The OVA appliance provides access to all of the vSphere Integrated Containers components.

The installation process involves several steps.

- Deploy the OVA, providing configuration information for vSphere Integrated Containers Registry and vSphere Integrated Containers Management Portal. The OVA deploys an appliance VM that provides the following services:
  - Runs vSphere Integrated Containers Registry
  - Runs vSphere Integrated Containers Management Portal
  - Makes the vSphere Integrated Containers Engine binaries available for download
  - Hosts the vSphere Client plug-in packages for vCenter Server
- Run the scripts to install the vSphere Client plug-ins on vCenter Server.
- Run the command line utility, vic-machine, to deploy and manage virtual container hosts.

For a full walkthrough of the installation process, please refer to the video below!
Creating a Virtual Container Host

The next step is to create a Virtual Container Host. This object has two components: A resource pool in which all the VMs corresponding to containers will reside and the container host itself, which is simply a VM running in the specified resource pool. The Virtual Container Host VM is responsible for implementing the Docker API and also acts as a routing endpoint for incoming traffic to its containers.
The command for creating the container host is rather long, so we have put it in a shell script file in order to simplify the lab a bit. Let's have a look at it!

1. If you are not already logged in to Docker CLI, double-click the Docker CLI icon on the desktop.
2. At the login prompt, type `root`.
3. Type the following command and press Enter:
   ```bash
cd /root/vic
   ```
4. Type the following command and press Enter:
   ```bash
cat create-demo.sh
   ```
5. Let's look at the flags!
   - `--name` - The name of the virtual container host
   - `--target` - The vCenter
   - `--user/--password` - Credentials for vCenter
   - `--volume-store` - Datastore where volumes belonging to the container host will be stored
   - `--image-store` - Datastore where container images will be stored
   - `--compute-resource` - vSphere cluster where the container host will run
   - `--public-network` - Network for publicly mapped ports in containers
   - `--bridge-network` - Private network for communication between containers
6. Now that we understand it a bit better, we can run the script. Type the following and press Enter:
   ```bash
   ./create-demo.sh
   ```
7. This will create the Virtual Container Host. This may take a few minutes.

**Verifying the Installation**

1. Verify that there were not errors. In the output from the previous command. Locate the line that says `DOCKER_HOST=192.168.100.x`. Take note of this address. This is the address to the Virtual Container Host.

2. Set the `DOCKER_HOST` variable. This makes all subsequent command interact with the container host we just created. Type the following and press Enter: (You will have to replace the "x" according to what the command output listed as the `DOCKER_HOST`)

   ```
   export DOCKER_HOST=192.168.100.x
   ```

3. Type the following command and press Enter:

   ```
   docker info
   ```

4. Notice how the server information indicates that we are now interacting with a vSphere cluster.
Cleaning up

In the subsequent exercises, we are going to use a pre-created Virtual Container Host, so we can delete the one we just created.

1. Type the following and press Enter:
   ```bash
   ./delete-demo.sh
   ```
2. Type the following and press Enter:
   ```bash
   unset DOCKER_HOST
   ```
Conclusion

In this module, we have discussed the following:

- The basic components of vSphere Integrated Containers
- How to install it
- How to perform basic configuration tasks.

You've finished Module 2

Congratulations on completing Module 2.

Proceed to any module below which interests you most.

- Module 3: Using vSphere Integrated Containers (15 minutes)
- Module 4: Introduction to Harbor (15 minutes)
- Module 5: Introduction to Admiral (15 minutes)

How to End Lab

To end your lab click on the END button.
Module 3: Using vSphere Integrated Containers (15 minutes)
Make Sure All Components are Running

Due to timing issues in the lab affecting the startup order of components, it is advisable to restart the containers that run the local Docker registry. You only need to do this once per lab, so if you have already done this as part of another module, you may skip this step. Should you experience problems with the registry refusing to authenticate you or timing out, you can always issue this command again.

Log in to Docker CLI

1. On the desktop, double-click the Docker CLI icon.
2. Log in with root and press the Enter key. You should be automatically logged.
Make Sure All Components are Running

1. At the command prompt, enter the following and press enter:

   ```
   ssh -o "StrictHostKeyChecking no" root@192.168.100.102 "systemctl restart docker"
   ```

2. When prompted for a password, use **VMware1**!

   It will take a minute or two, but you should be brought back to the command prompt.
Deploying a simple container

In this lesson, we will take a quick tour of the vSphere Integrated containers and see how it interoperates seamlessly with standard Docker commands and tools. Specifically, we will learn how to:

- Set up our shell to point to the Virtual Container Host
- Start a container on vSphere Integrated Containers
- Inspect the container through vCenter and perform some operations on it

Opening the command window

1. Open the console window by double-clicking the Docker CLI icon on the desktop.
2. When prompted for "login as", type root followed by Enter.

Setting the Docker host

This step is extremely important! Before we do anything else, we need to tell our local Docker command line tool that we are going to use an external container host (the Virtual Container Host). We can either specify the host on every command we type, but a much easier way is to use the DOCKER_HOST environment variable.

1. Type the following command and press Enter:
   ```bash
   export DOCKER_HOST=192.168.100.105:2375
   ```
Starting a Container

Once we have verified the installation, we can go ahead and start our first container. We're going to use busybox, which is a small Linux environment typically used for smaller devices, such as cable modems and home routers.

1. Let's go ahead and run it! Type the following and press Enter:
   ```bash
docker run -it 192.168.100.102/library/busybox
   ```
   Running this in a freshly installed container host may take a couple of minutes, so have patience. Subsequent invocations will be faster.

2. Once you get a prompt again, you should be inside the busybox container. Try a few commands, for example an `ls`. Type the following and press enter:
   ```bash
   ls
   ```

You may have noticed the IP address at the beginning of the image name above. This points to our local registry. We're using a local registry for two reasons: 1) The lab doesn't have Internet access, so pulling images from a public registry would fail and 2) we want to demonstrate the Harbor registry. More about that later. For now it's sufficient to say that you're using Harbor registry behind the scenes.

And what's a registry, you ask? It's simply the index and storage for Docker images.
Open a browser

1. Select **Chrome** from the start menu
A browser window will pop up.

Let's open vCenter and check what our operations resulted in! But first we need to log in to vCenter.

1. Once the window opens, click **vCenter (HTML5)** on the toolbar.
2. Check the **Use Windows session authentication** checkbox.
3. Click the **Login** button.

**Adjusting the browser zoom**

Depending on your screen, the resolution in the lab window may be too low to display some pages without cutting them off. If this happens, you need to change the zoom in your browser. A recommended zoom that will work on most devices is 80-90%.
1. At the top right corner, click on the three dots as shown above. A settings dialog will pop up.

![Settings dialog with zoom option](image)

1. Use the + and - buttons as shown above to adjust the zoom. An 80-90% usually works best.
A look under the hood

Once vCenter is open, we can have a look at the container we just created from a vCenter perspective. Remember that all vSphere Integrated Containers become virtual machines, so we should expect to find a corresponding machine for the container we just created.

1. Select the Hosts and Clusters icon from the navigation three.
2. Expand the `vcsa-01a.corp.local` vCenter
3. Expand the `RegionA01` data center
4. Expand the `RegionA01-COMP01` cluster
5. Expand the `vlab-vch-01` resource pool. This is the resource pool created along with the Virtual Container Host
6. You should now see two virtual machines. The one that's called `vlab-vch-01` is the controller host itself. The other virtual machine will have a auto generated name. This is the virtual machine that corresponds to your container. Select this virtual machine.
7. Take a look at the properties. You will notice, for example that it is connected to an internal bridge network with a 172.x.x.x address.
Perform a vMotion

Since the container we just started is implemented as a virtual machine, we can perform all operations we would normally use for a virtual machine. For example, we could do a vMotion to another host. Let's try that!

1. Right-click on the virtual machine corresponding to the container
2. Select Migrate...

[Diagram showing vMotion process]
1. Click the **Change compute resource only** radio button and
2. Click the **Next** button (not shown)

1. Select one of the hosts `esx-01a.corp.local` through `esx-03a.corp.local` and click the **Next** button
2. Click the **Next** button
3. Click the **Next** button
4. Click the **Finish** button

1. Once the migration has finished, you should see a different host in the virtual machine properties.
Clean up

This concludes this section. Let's exit the container we just created.

1. Exit the container shell by typing the following and pressing Enter:

   `exit`
Deploying an Application

In this lesson we will deploy a more realistic application. To demonstrate relationship between containers, we will deploy a two-tier application with a clustered database. The application is a simple product-rating function that stores its data in a redis database. To demonstrate clustering, we will store the data in the slave redis node and read it from the master node.

Structure of the application

In this section, we will deploy a more meaningful application. We’re going to build a simple product rating system for a storefront website. The customer feedback is stored in a replicated Redis datastore and accessed through a PHP-based front end. We are going to deploy each of these three components in its own container and make them all communicate on a private container network. The PHP frontend will expose port 80 to the public network.

One of the challenges we’re going to face is how to deploy a multi-tier application without having to manually deploy each tier and carefully tying them together. Luckily, there are several tools for doing that in an enterprise environment. Kubernetes is probably the most well-known one in that space.
In this exercise, we're going for a simpler and more lightweight solution using Docker Compose. This tool lets us define the container making up the tiers of the application in a single YAML file and bring the entire application up using a single command.

Examining docker-compose.yml

1. If you are not already logged into the Docker CLI, double-click the Docker-CLI icon on the desktop and log in as root (no password needed)
2. Type the following two commands, each followed by Enter:
   ```bash
cd /root/vhobby
cat docker-compose.yml
```
3. You should see a short text file. Let's examine the contents!

- **services** - Lists the services (containers) the application will spin up.
- **web** - This is our first service that we call "web" which will implement the web frontend.
- **links** - Specifies our dependencies. We are dependent on both the master and the slave redis nodes.
- **image** - The name of the docker image to spin up when the service starts.
- **ports** - Any ports we want to expose publicly. In this case, we're mapping the internal port 80 to port 80 on the public network.
- **redis-master, redis-slave** - Service definitions for the redis nodes.

Since we have dependencies between the nodes, Docker will automatically place them on a container network and link them together. It will also set up a simple naming service so that the code in the frontend can refer to the nodes by name instead of using IP addresses.
Deploying using docker-compose

1. Type the following and press Enter:
   ```bash
cd /root/vhobby
   ```

2. Let's make sure we're pointing to the Virtual Docker Host. Type the following and press Enter:
   ```bash
   export DOCKER_HOST=192.168.100.105:2375
   ```

3. User docker-compose to bring up the application. Type the following and press Enter:
   ```bash
docker-compose up -d
   ```

4. This may take a couple of minutes. You should not see any error messages.
Using the application

At this point, the application should be up and running. Let's try to log in to it!
1. From the start menu, select **Chrome**.

![Image of a website interface with a product rating page]

1. In the address field, type **192.168.100.105** followed by Enter.
2. You should now see a simple product rating page. Try entering a few ratings. The ratings are stored in the redis master and picked up from the slave, so this will test all three nodes of the application.
Log in to vCenter

Let's open vCenter and check what our operations resulted in! But first we need to log in to vCenter.
1. Select **Chrome** from the start menu.

1. Once the windows opens, click **vCenter (HTML5)** on the toolbar.
2. Check the **Use Windows session authentication** checkbox.
3. Click the **Login** button.

**Adjusting the browser zoom**

Depending on your screen, the resolution in the lab window may be too low to display some pages without cutting them off. If this happens, you need to change the zoom in your browser. A recommended zoom that will work on most devices is 80-90%.
1. At the top right corner, click on the three dots as shown above. A settings dialog will pop up.

[Diagram of settings menu with zoom options highlighted]

1. Use the + and - buttons as shown above to adjust the zoom. An 80-90% usually works best.
A look under the hood

Once vCenter is open, we can have a look at the container we just created from a vCenter perspective. Remember that all vSphere Integrated Containers become virtual machines, so we should expect to find a corresponding machine for the container we just created.

1. Select the Hosts and Clusters icon from the navigation three.
2. Expand the vcsa-01a.corp.local vCenter
3. Expand the RegionA01 data center
4. Expand the RegionA01-COMP01 cluster
5. Expand the vlab-vch-01 resource pool. This is the resource pool created along with the Virtual Container Host. You should now see four virtual machines. The one that's called vlab-vch-01 is the controller host itself. The other machines will have the vhobby prefix.
6. Click on the machine with a name starting with vhobby_web_1... and look at the properties. Notice that the virtual machine is only connected to the bridge network even though we exposed a port on the public network. This is because the routing of public addresses is done by the Virtual Container Host.
Clean up

1. Type the following and press Enter:
   ```bash
docker-compose down
   ``
   Due to timing issues in the HOL environment, you may see a timeout error. If that occurs, simply run the same command again.

2. Type the following and press Enter:
   ```bash
   /root/cleanup.sh
   ```

Using volumes

So far, our application only keeps persistent data as long as the containers are running. If we shut them down and start a new set of containers, the data will be gone.

To get around this problem, we can create external volumes where we keep the data and link them to our containers. As a first step, let’s create the containers. Remember that since we are pointing to the Virtual Container Host, the volumes will be created as files on the data store we specified when setting up the Virtual Container Host.

1. Create the redis slave volume. Type the following and press Enter:
   ```bash
docker volume create --name redis_slave
   ```

2. Create the redis master volume. Type the following and press Enter:
   ```bash
docker volume create --name redis_master
   ```
Volumes in the docker-compose file

Let's have a look at how to use the volumes we just created for our application. Since we're using docker-compose to bring up the application, we need to specify the volumes in the docker-compose.yml file. Again, since vSphere Integrated Containers integrates seamlessly with all docker tools, this is very straightforward. Let's look at the file!

1. Go to the directory where the persistent volume version of the app is stored. Type the following and press Enter:
   
   ```
   cd /root/vhobby/vol-vic
   ```

2. Examine the docker-compose file. Type the following and press Enter:
   
   ```
   cat docker-compose.yml
   ```

3. Let's examine the file. Notice how each of the two redis nodes are referring to their own volumes. Also notice at the very end of the file where we're tying the two volumes to the vsphere driver. That's it!
Bringing up the application with external volumes

1. Type the following command and press Enter:
   `docker-compose up -d`

We start the application using the normal docker-compose command.

1. Type `192.168.100.105` in the browser address field and press enter (not shown)
2. Open the application and enter a product review and click the `Submit` button (not shown)
3. Shut down the application. Type the following command and press Enter:
   `docker-compose down`
4. Start the application again by typing the following and pressing Enter:
   `docker-compose up -d`
5. Reload the web browser. Your product review should still be there. (Not shown)
Clean up

1. Type the following and press Enter:

   `docker-compose down`

2. Due to timing issues in the HOL environment, you may see a timeout error. If that occurs, simply run the same command again.

3. Type the following and press Enter:

   `/root/cleanup.sh`
Conclusion

In this module, we have discussed the following:

- Basic concepts of the Harbor Enterprise Registry
- A quick tour through the Harbor UI.

You've finished Module 3

Congratulations on completing Module 3.

Proceed to any module below which interests you most.

- Module 1: Introduction to vSphere Integrated Containers (15 minutes)
- Module 2: Setting up vSphere Integrated Containers (15 minutes)
- Module 4: Introduction to Harbor (15 minutes)
- Module 5: Introduction to Admiral (15 minutes)

How to End Lab

To end your lab click on the END button.
Module 4: VMware Harbor
- An Enterprise-class Registry Server (15 minutes)
Introduction to Registry Servers and VMware Harbor

What is a Docker Registry?

Registries are servers that provide a central content storage and distribution system for Docker images and are focused on delivering a way to reliably manage, package, and exchange content. They present an HTTP based API for managing the transfer of Docker images to a local machine using the `docker pull` and `docker push` commands. Registry servers are organized by collections of Repositories - which are a collection Docker images. Registries support image tagging to make categorization and identification within a repository easier.

Registries may be privately or publically hosted. By default, all docker deployments have the ability to pull content from Docker's public registry instance and to push content using a Docker Hub account. Although public registries like this often have a large number of available images to consume, they contribute little oversight or image verification, making security in an enterprise environment challenging. Individuals and enterprises needing a higher level of security when managing private images and content often need deploy a private registry server for protecting sensitive images that require a higher level of governance. The open source Docker Distribution project provides a private registry implementation with basic functionality, but it is still missing many critical enterprise-class features.

Introducing VMware Harbor

Project Harbor™ is an enterprise-class registry server that stores and distributes Docker images. Harbor™ extends the open source Docker Distribution by adding functionality usually required by an enterprise, such as security, identity and management. As an enterprise private registry, Harbor™ offers better performance and security. Having a registry closer to the build and run environment improves the image transfer efficiency. Harbor™ supports the setup of multiple registries and has images replicated between them. With Harbor™, the images are stored within the private registry, keeping the bits...
and intellectual properties behind the company firewall. In addition, Harbor™ offers advanced security features, such as user management, access control and activity auditing.

**Harbor's Enterprise-class Registry Server Features**

- **Role Based Access Control** - Users and docker repositories are organized via "projects", a user can have different permission for images under a namespace.
- **Image Replication** - Images can be replicated (synchronized) between multiple registry instances. Great for load balancing, high availability, hybrid and multi-cloud scenarios.
- **Graphical User Portal** - User can easily browse, search docker repositories, manage projects/namespaces.
- **AD/LDAP Support** - Harbor™ integrates with existing enterprise AD/LDAP for user authentication and management.
- **Auditing** - All the operations to the repositories are tracked and can be used for auditing purpose.
- **Internationalization** - Already localized for English, Chinese, German, Japanese and Russian. More languages can be added.
- **RESTful API** - RESTful APIs are provided for most administrative operations of Harbor™. The integration with other management softwares becomes easy.
- **Easy Deployment** - Provide both an online and offline installer. Besides, a virtual appliance for vSphere platform (OVA) is available.
Role based access control (RBAC)

A unique feature of the Harbor registry is the security and role based access control features. This allows an organization to define and delegate responsibilities in a fine-grained and flexible manner, while keeping strict control of access rights and security.

Harbor manages images through projects. Users can be added into one project as a member with three different roles:

- **Guest**: Guest has read-only privilege for a specified project.
- **Developer**: Developer has read and write privileges for a project.
- **ProjectAdmin**: When creating a new project, you will be assigned the "ProjectAdmin" role to the project. Besides read-write privileges, the "ProjectAdmin" also has some management privileges, such as adding and removing members.

Besides the above three roles, there are two system-wide roles:

- **SysAdmin**: "SysAdmin" has the most privileges. In addition to the privileges mentioned above, "SysAdmin" can also list all projects, set an ordinary user as administrator and delete users. The public project "library" is also owned by the administrator.
- **Anonymous**: When a user is not logged in, the user is considered as an "Anonymous" user. An anonymous user has no access to private projects and has read-only access to public projects.
Identity stores

Harbor supports two types of identity stores:

**Database (local)**

This is the simplest way of implementing access control in Harbor. Users and their credentials are stored in a local database that's managed by Harbor itself. This mode has three distinct advantages:

- Support for self-registration. In an open registry, it may be beneficial to allow new users to create their own user accounts without the intervention from an administrator. If enabled, the database identity store allows this type of registration.
- Self-service for forgotten passwords.
- Ease of implementation since there are no additional components that need to be managed.

**LDAP/AD**

In this mode, Harbor uses an external LDAP-compatible identity store, such as Microsoft Active Directory. This is typically the preferred type of identity store for a proprietary registry in a larger enterprise. It allows central management of the identity store and an additional layer of security. However, it doesn't support self-registration and password reset.

Managing projects

A project in Harbor contains all repositories of an application. No images can be pushed to Harbor before the project is created. RBAC is applied to a project. There are two types of projects in Harbor:

- **Public**: All users have the read privilege to a public project, it's convenient for you to share some repositories with others in this way.
- **Private**: A private project can only be accessed by users with proper privileges.

Replication

Image replication is used to replicate repositories from one Harbor instance to another.

The function is project-oriented, and once the system administrator set a rule to one project, all repositories under the project will be replicated to the remote registry. Each repository will start a job to run. If the project does not exist on the remote registry, a new project will be created automatically, but if it already exists and the user configured in policy has no write privilege to it, the process will fail. When a new repository is pushed to this project or an existing repository is deleted from this project, the same
operation will also be replicated to the destination. The member information will not be replicated.

There may be a bit of delay during replication according to the situation of the network. If replication job fails due to the network issue, the job will be re-scheduled a few minutes later.
Make Sure All Components are Running

Due to timing issues in the lab affecting the startup order of components, it is advisable to restart the containers that run the local Docker registry. You only need to do this once per lab, so if you have already done this as part of another module, you may skip this step. Should you experience problems with the registry refusing to authenticate you or timing out, you can always issue this command again.

Log in to Docker CLI

1. On the desktop, double-click the Docker CLI icon.
2. Log in with root and press the Enter key. You should be automatically logged.
Make Sure All Components are Running

1. At the command prompt, enter the following and press enter:

```bash
ssh -o "StrictHostKeyChecking no" root@192.168.100.102 "systemctl restart docker"
```

2. When prompted for a password, use **VMware1!**

   It will take a minute or two, but you should be brought back to the command prompt.
A Quick Tour of the Harbor UI

In this lesson, we will explore the Harbor User Interface and how it can be used to manage registries. Specifically, we will cover the following:

- A look at projects
- How to push a repository to Harbor
- What a repository looks like in Harbor
- Pulling images from Harbor
- Replication of registries

Logging in
1. From the start menu, select **Chrome**

![Chrome window](image)

1. Click the **vSphere Integrated Containers** icon on the toolbar
2. Enter username **admin**
3. Enter password **VMware1!**
4. Click the **LOG IN** button.

**Adjusting the browser zoom**

Depending on your screen, the resolution in the lab window may be too low to display some pages without cutting them off. If this happens, you need to change the zoom in your browser. A recommended zoom that will work on most devices is 80-90%.
1. At the top right corner, click on the three dots as shown above. A settings dialog will pop up.

1. Use the + and - buttons as shown above to adjust the zoom. An 80-90% zoom usually works best.

**Looking at projects**
Repositories in harbor are organized into projects. Each project typically maps to some organizational unit and can define its own access rules.

1. Click on the **Registry** tab.
2. Click on the **apps** project.

**Exploring the apps project**

1. Click on the **apps/vhobby** repository.

<table>
<thead>
<tr>
<th>Name</th>
<th>Tags</th>
<th>Pulls</th>
</tr>
</thead>
<tbody>
<tr>
<td>apps/vhobby</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>apps/redis</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>apps/redis-slave</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>apps/redis-master</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

1. Examine the metadata and the **pull command**.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Pull Command</th>
<th>Signed</th>
<th>Author</th>
<th>Creation Time</th>
<th>Docker Version</th>
<th>Architecture</th>
<th>OS</th>
</tr>
</thead>
</table>
The pull command is what you would use on a Docker command line to download the repository.

**Opening the command window**

1. Open the console window by double-clicking the **Docker CLI** icon on the desktop.
2. When prompted for "login as", type **root** followed by Enter.

**Pushing a repository to Harbor**

Let's store an new image in the registry! We're going to build a simple "Hello World!" application and push it to the Harbor registry.

1. Go to the directory of the Dockerfile by typing the following and pressing Enter:
   
   ```bash
   cd /root/hello
   ```
2. Make sure we're using the local Docker host. Type the following and press Enter:
   ```bash
   unset DOCKER_HOST
   ```
3. Build the image and apply repository tag. Type the following and press Enter. **Notice the dot at the end!**
   ```bash
   docker build -t 192.168.100.102/library/hello .
   ```
4. Finally, push the repository to Harbor. Type the following and press Enter:
   ```bash
   docker push 192.168.100.102/library/hello
   ```

## Looking at the repository in Harbor

1. Go back to the **Chrome** window (not shown)
2. Click the **Registry** tab and navigate to the **library** project (not shown)
3. Click on the repository named **library/hello**.

### library/hello

<table>
<thead>
<tr>
<th>Tag</th>
<th>Pull Command</th>
<th>Signed</th>
<th>Author</th>
<th>Creation Time</th>
<th>Docker Version</th>
<th>Architecture</th>
<th>OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>latest</td>
<td><code>docker pull 192.168.100.102:443/library/hello:latest</code></td>
<td></td>
<td></td>
<td>6/6/2017, 8:42 AM</td>
<td>1.12.1</td>
<td>amd64</td>
<td>linux</td>
</tr>
</tbody>
</table>
1. Examine the repository and take note of the metadata and pull command.

**Verify that we can pull the repository**

![Image showing commands to remove and pull the repository]

1. Remove the local copy of the image. Type the following and press Enter:
   
   ```
   docker rmi 192.168.100.102/library/hello
   ```

2. Pull the repository from Harbor
   
   ```
   docker pull 192.168.100.102/library/hello
   ```

**Replication**

![Image showing replication configuration in vSphere Integrated Containers]

1. Replication
2. Endpoints
3. Filter Endpoints
1. Click on the **Replication** link.
2. Select the **Endpoints** tab.
3. Click on the **+ENDPOINT** link

1. This is where you would specify a replicated registry. In this lab there is only one repository, so close this Window by clicking the **Cancel** button.
Projects, users and roles

In this lesson, we will look closer at user, project and roles as well as how they interact to create a secure but flexible environment. We will learn the following:

- Creating users
- Creating projects
- Assigning roles
- Using the registry with a specific user

Create a user

Let's create some users for the subsequent exercises. We will create two users:

- Sally Superbrain - A developer who will be given read/write access to our new project.
- Terry Testy - A tester who will only have read access to our new project.

1. Click on the **Registry** tab.
2. Select **Users** from the left-hand menu.
3. Click **+USER** to add a new user.
Enter user details

1. Enter username **sally**
2. Enter email **sally@corp.local**
3. Enter First and last name **Sally Superbrain**
4. Enter password **VMware1!**
5. Confirm password **VMware1!**
6. Click **OK**
Create a tester user

Username: terry
Email: terry@corp.local
First and last name: Terry Testy
Password: VMware1!

Repeat the same process as above and create a user with the following attributes.

1. Username **terry**
2. Email **terry@corp.local**
3. First and last name **Terry Testy**
4. Password **VMware1!**
Create a new project

Projects are containers for repositories attached to a real-life project or some other organizational unit. They are used to create a common role-based access model around a set of repositories. Let's create a project!

1. Click the **Registry** tab
2. Select **Projects** from the left-hand menu
3. Click **+PROJECT**
Providing details for the new project

Let's create a new project called "hol". We're going to create it as a private project, meaning that only authenticated users can pull from it.

1. Enter project name **hol**
2. Leave the **Public** checkbox unchecked
3. Click **OK**

Assign users and roles

Let's assign some users to the project and give them roles determining what they're allowed to do.
1. Click on the **hol** project.

![Project Admin interface with Members tab highlighted]

1. Click on the Members tab
2. Click on +MEMBER

**Assign a developer to the project**

![New Member interface with name and role fields]

1. Enter name **sally**
2. Click the Developer role
3. Click OK
Assign a guest role for the tester

Assign another user by repeating the same process as above. Fill in the following:

1. Name: terry
2. Role: Guest
Push a repository

Let's tag and push a repository to our new project. To do that, we need to log in as a developer user, which is Sally in our case.

1. To log in, type the following and press Enter after each line
   
   ```bash
   docker login 192.168.100.102
   sally
   VMware1!
   ```

2. Let's tag an image to prepare it to be pushed to the registry. Type the following and press Enter after each command:

   5. `docker tag busybox 192.168.100.102/hol/busybox`
   6. `docker push 192.168.100.102/hol/busybox`
Pull a repository as a read-only user

Let's log in as Terry and try to pull a repository from Harbor. However, Terry is trying to break the rules. Let's see if Harbor can prevent him from doing that!

1. Log in as Terry by typing the following and pressing Enter:
   ```bash
docker login -u terry -p VMware1! 192.168.100.102
   ```

2. Try to do something we're not allowed to. Let's try to push the buybox registry to the hol project!
   ```bash
docker push 192.168.100.102/hol/busybox
   ```

3. As you can see, we're not allowed to do that. Try to pull it instead. Type the following and press Enter:
   ```bash
docker pull 192.168.100.102/hol/busybox
   ```

Inspect the repository in Harbor

Finally, let's check what the repository looks like in Harbor.
1. Click the **Registry** tab
2. Click **Projects** in the left-hand menu
3. Select the **hol** project
1. Click the **hol/busybox** repository

![Repository Table]

1. Examine the repository

Finally, let's delete the repository. We can do that, since we're logged in as admin and have the project administrator role.

![Delete Repository Dialog]

1. Click the three dots next to the repository
2. Select Delete
3. Click OK in the dialog box (not shown)
Conclusion

In this module, we have discussed the following:

• Basic concepts of the Harbor Enterprise Registry
• A quick tour through the Harbor UI.

You've finished Module 4

Congratulations on completing Module 4.

Proceed to any module below which interests you most.

• Module 1: Introduction to vSphere Integrated Containers (15 minutes)
• Module 2: Setting up vSphere Integrated Containers (15 minutes)
• Module 3: Using vSphere Integrated Containers (15 minutes)
• Module 5: Introduction to Admiral (15 minutes)

How to End Lab

To end your lab click on the END button.
Module 5: Container Management with VMware Admiral (15 minutes)
An Introduction to VMware Admiral

In this lesson, we'll take a tour through Admiral, the light-weight, fast and feature rich container management platform from VMware.

NOTE: This section is all reading and does not contain any interactive content. We strongly recommend you study this section, since it introduces some important concepts that will be used throughout the interactive portion of the lab.

A Introduction to VMware Admiral

While VMware Harbor provides a safe and feature-rich location to store your fleet of containers, VMware Admiral provides the robust management capabilities needed to direct that fleet.

Admiral is a highly scalable and very lightweight Container Management platform for deploying and managing container based applications. Its capabilities include modeling, provisioning and managing containerized applications via a UI, yaml-based template or Docker Compose file, as well as configuring and managing container hosts. Although most developers will continue to interface with VIC, Admiral, and Harbor through the APIs and Docker command line, not all Enterprise IT operations staff have experience with those or may need an easier way to manage large collections of applications.

A main feature of the Admiral project is the ability to provision containerized applications. Admiral makes use of the Docker Remote API to provision and manage containers, including retrieving stats and info about container instances.

Developers will use Docker Compose, Admiral templates or the Admiral UI to compose their application and deploy it using Admiral provisioning and orchestration engine.

Administrators can manage container host infrastructure and apply governance to its usage, including grouping of resources, policy based placements, quotas and reservations and elastic placement zones.

Admiral, as well as Harbor (our enterprise class registry) are both included in the commercial distribution of vSphere Integrated Containers. You can find it under the Management tab in the vSphere Integrated Containers UI. Admiral is also utilized by vRealize Automation for container management capabilities.
Admiral Concepts

Before we start digging into the UI, let's briefly go over some of the main concepts in Admiral.

**Host** - This is simply a Docker host that's implemented either using native Docker or vSphere Integrated Containers.

**Placement** - An abstract grouping of Docker hosts that applications are deployed into. This concept is very similar to a vSphere Cluster. Placements have resource limits and a placement strategy controlling how workloads are distributed across hosts.

**Placement Zone** - A grouping of hosts associated with a placement.

**Template** - An application template created from a Dockerfile or using Admiral itself.

**Application** - A running instance of a Template.

**Placement and Host Selection**

Placement and host selection is a central concept in Admiral. An application can be deployed into different environments and each environment may be tied to a different set of resources. Admiral uses the project, placement zone and placement objects to implement this.

Let's look at a simple use case to illustrate the placement and allocation logic and concepts. Assume we have typical Dev/Stage/Production environments that we need to isolate and assign separate resources to. In our example, we have two placement zones only: One for prod and one for non-prod and we make both the test and the dev point to the same placement zone. To do that, we would create the following objects in Admiral:

1. **Projects** - Create three separate projects, one for each environment.
2. **Placement Zones** - Create placement zones and assign resources to each one. We may create a placement zone for each project, or two or more project may draw resources from the same placement zone.

3. **Placements** - Provides a link from a project to a placement zone. In our example, Dev and Test would point to the Non-prod placement zone, whereas the Prod project would point to the Prod placement zone.

Once this configuration is in place, we can deploy applications into a specific environment (e.g. "Dev") and the Placement Zone tied to the "Dev" project will determine which hosts will be chosen for the containers.

**Resource allocation**

In addition to acting like a link between projects and placement zones, a Placement also serves as a resource allocation construct. Each placement carries information about how much of the underlying resources to use. Let's take a look at the information held by a placement!

- **Priority** - If a project has more than one possible placement, this value determines the order in which placement groups are selected.
- **Instances** - (Integer > 0) Maximum number of containers that can be provisioned.
- **Memory Limit** - A number between 0 and the memory available in the placement zone. This is the total memory available for resources in this placement. 0 means no limit. If not zero, next minimum is 4M.
- **CPU Shares** - The provisioned resource will be given this amount of CPU shares (relative weight). All containers in this placement zone will be configured with that number of CPU shares. By default, all containers get the same proportion of CPU cycles. This proportion can be modified by changing the containers CPU share weighting relative to the weighting of all other running containers.
Host selection based on tags

In addition to statically allocating a set of hosts to a placement zone, we can dynamically select host based on tags. For example, we could create a tag called "Production" and attach that to all production hosts. A placement zone may then refer to the hosts indirectly using that tag. The net effect of this is that this allows an administrator to dynamically add and remove hosts without having to change the placement zones. If we need to add more production hosts, all we need to do is to spin them up, register them and attach the production tag to them.
Make Sure All Components are Running

Due to timing issues in the lab affecting the startup order of components, it is advisable to restart the containers that run the local Docker registry. You only need to do this once per lab, so if you have already done this as part of another module, you may skip this step. Should you experience problems with the registry refusing to authenticate you or timing out, you can always issue this command again.

Log in to Docker CLI

1. On the desktop, double-click the Docker CLI icon.
2. Log in with root and press the Enter key. You should be automatically logged.
Make Sure All Components are Running

1. At the command prompt, enter the following and press enter:

   ```
   ssh -o "StrictHostKeyChecking no" root@192.168.100.102 "systemctl restart docker"
   ```

2. When prompted for a password, use **VMware1!**

   It will take a minute or two, but you should be brought back to the command prompt.
A Quick Tour Through the UI

In this lesson, we will try out the Admiral User interface and show how it is used to manage a container environment. Specifically, we will look at the following areas:

- Adding containers hosts
- Placement zones
- Placements
- Templates
- Deploying instances from templates

Introduction

Let's take a quick tour through the Admiral UI with the goal of creating an environment and a template for the product review application we've experimented with in previous sections.
Open the browser

1. Open Google Chrome from the Start menu.

2. Navigate to the URL: https://192.168.100.102/harbor/sign-in.
1. From the start menu, select **Chrome**.
2. From the Chrome toolbar, select **vSphere Integrated Containers**.

1. Enter username **admin**
2. Enter password **VMware1!**
3. Click the **LOG IN** button.
4. Click the **Management** tab.

Add a host

Before we can do anything, we need to add at least one host under management by Admiral. Host can be either traditional Docker hosts or Virtual Containers Hosts from...
vSphere Integrated Containers. In this exercise, we'll use a standard Docker host. Both types of hosts behave identically from the point of view of Admiral.

Before we can do anything else, we need to register the Docker hosts we're going to use.

1. Click the **ADD A HOST** button.
2. Enter address [https://192.168.100.104:2376](https://192.168.100.104:2376)
3. Select host type **Docker**
4. Select login credential **default-client-cert**.
5. Leave **Enable host auto configuration** unchecked.
6. Click the **VERIFY** button. You should see a message saying that the host was verified successfully.
7. Click the **ADD** button.
8. You should now see a tile representing the host.
Configure a placement zone

A placement zone is similar to a cluster. It's simply a collection of hosts we can draw capacity from. Let's create a placement zone and add our host to it!

1. Hover over the top right corner of the host. A menu will appear.
2. Click the edit icon (looks like a pencil).

Edit the host to add the placement zone
1. From the Placement zone drop-down, select **lab**.
2. Click the **UPDATE** button.
3. If the placement zone doesn't appear on the host tile, click the small refresh icon next right above the host tile.

## Create a placement

Placements are the links between projects and Placement Zones. You can think of them as a way of carving out capacity from a placement zone and assigning it to a project. Typically, you would specify limitations on how much resources a project can pull from the placement zone. In our case, however, we'll allocate the entire placement zone to this project.

1. From the left-hand menu, select **Placements**.
2. Click the **Add** icon on the right hand side.
3. Enter **vhobby-dev** in as placement name.
4. Select Project **vhobby dev**.
5. Select Placement zone **lab**. Leave all other fields empty
6. Click the **SAVE** button.
Create a template

At this point, we have a simple infrastructure we can start deploying applications into. Let's create an application template. Templates can be created either from native Admiral definition files or from Docker Compose files. In this example, we will use the Docker Compose file for the product review application we used in earlier exercises.

1. From the left-hand menu, select **Templates**
2. Click the **upload** icon (looks like an arrow pointing upwards)
Load the template content

The "source code" for a template is a docker-compose file. Docker-compose is a simple way of defining a multi-tier application. In our case, we'll upload a docker-compose file describing our Hobby Store application. It consists of a web front-end and two redis nodes for data storage.

1. Click the **LOAD FROM FILE** button. Navigate to `c:\labdata\vhobby\docker-compose.yml` and click **Open**. (Not shown)
2. Click the **IMPORT** button.

---

1. Click the **LOAD FROM FILE** button. Navigate to `c:\labdata\vhobby\docker-compose.yml` and click **Open**. (Not shown)
2. Click the **IMPORT** button.
Reviewing and renaming the template

At this point, you should see the contents of the template as defined by the Docker Compose file. You will see a web node and two redis nodes. The template will have an automatically generated name. Let's change it!

1. Click the edit icon (looks like a pencil) next to the template name.
2. Enter vhobby as the template name.
3. Click the checkmark icon
Deploy an instance of the application

Now that we have a host, a placement zone, a placement and a template, we can do ahead and deploy and application.

Let's deploy an instance of the vhobby application!

1. Select **Templates** from the left-hand menu.
2. In the search field, type **vhobby** followed by Enter. A single tile named vhobby should appear.
3. Click the **PROVISION** button. A placement zone selector will appear.
4. Select the **vhobby dev** placement zone.
5. Click the **PROVISION** button again.
6. You should see a progress bar on the right-hand side. It should turn green after a few seconds.
Inspecting the application

We should now have a running application. Let's have a look at it in Admiral!

1. From the left-hand menu, select Applications. You should see a single tile.
2. Click the containers icon (looks like a cube).
3. You should see three tiles representing the containers making up the application.
Inspecting the web node

- Policies
  - Placements

- Templates
  - Templates
  - Registries

- Deployments
  - Applications
    - Containers
    - Networks
    - Volumes

- Running for 2 hours

- Host: photon-machine
- Created: June 19, 2017 9:24 AM
- Ports:
  - http://192.168.100.104:80.80/tcp
- Command: apache2-foreground
1. On the web node (name starts with web-), hover over the right-hand corner and click the inspect icon (looks like an eye).

1. Review the fields on the screen, such as CPU and memory utilization.
2. Take note of the log output on the right-hand side.
3. Take note of the Ports field which will show the ports exposed by your application.
Testing the application

The port address above is where we should find the application running. Since we configured only a single host, we know that it's going to be 192.168.100.104. Let's try it!

1. Open a new browser tab.
2. Enter address 192.168.100.104 and press Enter.

You should now see the Hobby Shop application. Feel free to enter a few product reviews!
Clean up

1. Click on the browser tab for vSphere Integrated Containers (not shown)
2. From the left-hand side menu, select Applications.
3. On the vhobby application tile, hover over the right-hand corner and click the Remove icon (looks like a garbage can)
4. Confirm by clicking Delete on the pop up (not shown)
Conclusion

In this module, we have discussed the following:

- Basic concepts of Admiral
- A quick tour through the Admiral UI.

You've finished Module 5

Congratulations on completing Module 5.

Proceed to any module below which interests you most.

- Module 1: Introduction to vSphere Integrated Containers (15 minutes)
- Module 2: Setting up vSphere Integrated Containers (15 minutes)
- Module 3: Using vSphere Integrated Containers (15 minutes)
- Module 4: Introduction to Harbor (15 minutes)

How to End Lab

To end your lab click on the END button.
Conclusion

Thank you for participating in the VMware Hands-on Labs. Be sure to visit http://hol.vmware.com/ to continue your lab experience online.

Lab SKU: HOL-1830-02-CNA

Version: 20171129-153804