

# Photon Controller 1.1 User Guide

## Contents

<b>1</b>	<b>Photon Controller and Photon Platform</b>	<b>5</b>
1.1	Integration with VMware NSX . . . . .	5
<b>2</b>	<b>Quick-Start Installation</b>	<b>6</b>
2.1	Introduction . . . . .	6
2.1.1	Overview . . . . .	6
2.1.2	Lightwave Authentication . . . . .	6
2.1.3	Assumptions . . . . .	6
2.1.4	Requirements . . . . .	7
2.2	Installing the Photon Controller Installation OVA . . . . .	7
2.3	Installing the Photon Controller CLI on a Linux Workstation . . . . .	7
2.4	Preparing ESXi for Photon Controller . . . . .	8
2.5	Preparing a YAML Configuration File . . . . .	8
2.5.1	The Anatomy of a Configuration File . . . . .	8
2.5.2	Avoiding Errors in the Configuration File . . . . .	9
2.5.3	Example Basic Configuration File . . . . .	11
2.6	Deploying the Basic System . . . . .	11
2.7	Checking the System's Status . . . . .	11
2.8	Redeploying with Lightwave Authentication . . . . .	12
2.8.1	Destroying the Previous Deployment . . . . .	12
2.8.2	Preparing ESXi for Authentication . . . . .	12
2.8.3	Adding Lightwave Authentication to the Template . . . . .	13
2.8.4	Example Configuration File with Authentication . . . . .	14
2.8.5	Building Your Configuration File and Deploying Photon Controller . . . . .	15
2.9	Creating Accounts in Lightwave . . . . .	16
2.10	Connecting to the Load Balancer . . . . .	16
2.11	Creating a Kubernetes Cluster . . . . .	17
2.12	Troubleshooting . . . . .	17
2.12.1	Log Files for the Installer . . . . .	17
2.12.2	Deploying the OVA Installer . . . . .	17
2.12.3	Lightwave Authentication and NTP . . . . .	18
<b>3</b>	<b>Authentication and Authorization</b>	<b>18</b>
3.1	Authenticating Multitenant Users and Groups . . . . .	18
3.1.1	Lightwave Security Services . . . . .	18
3.1.2	Roles and Rights . . . . .	18
3.1.3	Process Overview . . . . .	19
3.1.4	Connecting to the Lightwave Management VM . . . . .	19
3.1.5	Gaining Access to the Docker Container Running Lightwave . . . . .	19
3.1.6	Creating a System Administrator in Lightwave . . . . .	20
3.1.7	Creating a Tenant Administrator . . . . .	20
3.1.8	Creating Project Users . . . . .	20

3.1.9	Using Lightwave Groups in Photon Controller . . . . .	20
3.1.10	Setting Security Groups for Administrators, Tenants, and Project Users . . . . .	21
3.1.11	Connecting to the Load Balancer for Secure Login . . . . .	21
3.1.12	Tenants, Resource Tickets, and Projects . . . . .	22
3.1.13	Projects . . . . .	23
3.2	The Authorization Model . . . . .	23
3.2.1	System Administrator . . . . .	23
3.2.2	Tenant Administrator . . . . .	24
3.2.3	Project Users . . . . .	24
<b>4</b>	<b>Basic Operations</b>	<b>24</b>
4.1	Connecting the Photon Load Balancer or Management VM . . . . .	24
4.2	Logging In to Photon Controller . . . . .	25
4.3	Connecting to the Web UI . . . . .	25
4.4	An Overview of Photon Commands . . . . .	27
4.4.1	Setting Targets and Logging In . . . . .	27
4.4.2	Images . . . . .	28
4.4.3	Tenants and Projects . . . . .	28
4.4.4	Flavors and Disks . . . . .	28
4.4.5	Virtual Machines and Disks . . . . .	29
4.4.6	Clusters . . . . .	30
4.4.7	Availability Zones . . . . .	30
4.4.8	Authentication . . . . .	30
4.5	Working with ESXi Hosts . . . . .	31
4.5.1	Adding an ESXi Host to a Photon Controller Cluster . . . . .	31
4.5.2	Viewing Information about ESXi Hosts . . . . .	33
<b>5</b>	<b>Working with Tenants, Projects, and VMs</b>	<b>34</b>
5.1	About Tenants, Resource Tickets, and Projects . . . . .	34
5.1.1	Tenants . . . . .	34
5.1.2	Resource Tickets . . . . .	35
5.1.3	Projects . . . . .	35
5.2	Working with Tenants . . . . .	35
5.2.1	Creating a Tenant . . . . .	35
5.2.2	Commands for Working with Tenants . . . . .	36
5.2.3	Related API Endpoints . . . . .	36
5.3	Creating a Project . . . . .	36
5.3.1	Project Commands . . . . .	36
5.3.2	Example of How To Create a Tenant and a Project . . . . .	37
5.4	Uploading Images . . . . .	37
5.4.1	Setting the Target . . . . .	38
5.4.2	Uploading Images . . . . .	38
5.4.3	Eager Images and On-Demand Images . . . . .	38
5.4.4	Viewing Images . . . . .	38
5.4.5	Deleting Images . . . . .	38
5.5	Creating Images . . . . .	38
5.5.1	Creating Images . . . . .	39
5.5.2	Viewing the List of Images . . . . .	39
5.5.3	Uploading an OVA File to Create an Image . . . . .	39
5.5.4	Creating an Image from a VM . . . . .	39
5.5.5	Deleting an Image . . . . .	39
5.6	Replicating Images in Datastores . . . . .	40
5.6.1	Image Datastores . . . . .	40
5.6.2	Image Replication and Datastores . . . . .	40

5.7	Creating Flavors . . . . .	40
5.7.1	Types of flavors . . . . .	40
5.7.2	Flavor Costs . . . . .	40
5.7.3	Creating Flavors . . . . .	41
5.7.4	List All Flavors . . . . .	41
5.7.5	Show Flavor Details . . . . .	42
5.7.6	Create Cluster with New Flavor . . . . .	42
5.8	Provisioning Virtual Machines . . . . .	42
5.8.1	Creating a Virtual Machine from an Image . . . . .	42
5.8.2	Getting Information about VMs . . . . .	43
5.8.3	Attaching Persistent Disks to VMs . . . . .	43
5.8.4	Attaching an ISO to a VM . . . . .	44
5.8.5	Operating a Virtual Machine . . . . .	44
5.9	Creating a Network . . . . .	44
5.9.1	Checking the Default Network . . . . .	44
5.9.2	Creating a New Network . . . . .	45
5.9.3	Setting the Default Network . . . . .	45
5.10	Using Photon OS . . . . .	47
5.10.1	Producing a Photon OS VM in Photon Controller . . . . .	47
5.10.2	Docker Containers on Photon OS . . . . .	50
<b>6</b>	<b>Deploying Clusters</b>	<b>50</b>
6.1	Creating a Kubernetes Cluster . . . . .	50
6.1.1	Introduction . . . . .	51
6.1.2	Requirements . . . . .	51
6.1.3	Obtaining, Uploading, and Enabling the Kubernetes Image . . . . .	51
6.1.4	Creating a Tenant and a Project for the Cluster . . . . .	52
6.1.5	Creating Resources for Use in the Cluster . . . . .	52
6.1.6	Creating Resources for Containerized Applications . . . . .	52
6.1.7	Setting Up a Network for Use with the Cluster . . . . .	53
6.1.8	Creating the Kubernetes Cluster . . . . .	53
6.1.9	Checking the Cluster's Resources and Status . . . . .	53
6.1.10	Opening the Kubernetes Dashboard . . . . .	57
6.1.11	Deploying an nginx Web Server . . . . .	57
6.1.12	Troubleshooting Cluster Creation . . . . .	63
6.1.13	Related . . . . .	64
6.2	Obtaining and Uploading Images for Clusters . . . . .	64
6.2.1	Downloading the Base Image for a Cluster . . . . .	64
6.2.2	Uploading Images to Photon Controller . . . . .	65
6.2.3	Enabling The Cluster Type for an Image . . . . .	65
6.3	Running Tomcat on a Kubernetes Cluster . . . . .	65
6.3.1	Spinning Up a Cluster for Tomcat . . . . .	65
6.3.2	Deploying an App to the Cluster . . . . .	65
6.3.3	Scaling with Kubernetes . . . . .	66
6.3.4	Dealing with Environment Size Limitations . . . . .	66
6.4	Managing Clusters . . . . .	67
<b>7</b>	<b>Information for Developers</b>	<b>67</b>
7.1	Setting Up Your Own Load Balancer . . . . .	67
7.1.1	Ports . . . . .	67
7.1.2	Headers . . . . .	68
7.2	Compiling the Command-Line Utility . . . . .	68
7.2.1	Compiling CLI from source . . . . .	68
7.2.2	Compatibility . . . . .	68

7.2.3	Testing Binary . . . . .	68
7.2.4	Hitting a Problem? . . . . .	69
7.3	Installing Photon Controller on Photon OS . . . . .	69
<b>8</b>	<b>Integrating VMware vSAN with Photon Platform</b>	<b>70</b>
8.1	Setting Up VMware vSAN . . . . .	70
8.1.1	Prerequisites . . . . .	70
8.1.2	Requirements . . . . .	70
8.2	Installing the vSAN Management Service . . . . .	71
8.2.1	Enabling SSL for vSAN . . . . .	71
8.3	Prepare ESXi Hosts for vSAN . . . . .	71
8.4	Setting Up vSAN for Photon Controller . . . . .	72
8.4.1	Connect to the vSAN Management VM . . . . .	72
8.4.2	Create a vSAN cluster . . . . .	72
8.4.3	Join the ESXi Hosts to the vSAN Cluster . . . . .	72
8.4.4	Enable auto claim disks on a host . . . . .	73
8.4.5	Auto claim disks on a host . . . . .	73
8.4.6	Check the vSAN Cluster . . . . .	73
8.5	Detecting the vSAN Datastore After Setup . . . . .	73
8.6	Removing the vSAN Datastore . . . . .	73
8.7	Scripting the Deployment . . . . .	74
<b>9</b>	<b>API</b>	<b>75</b>
9.1	Viewing the Interactive API Documentation . . . . .	75
9.2	Using the API . . . . .	77
9.2.1	The API . . . . .	77
9.2.2	API Port Numbers . . . . .	77
9.2.3	Authentication . . . . .	77
<b>10</b>	<b>Troubleshooting and Maintenance</b>	<b>79</b>
10.1	Maintaining Hosts . . . . .	79
10.1.1	Host States . . . . .	80
10.1.2	Suspending the Host and Entering Maintenance Mode . . . . .	80
10.2	Troubleshooting Installation and Operations . . . . .	80
10.2.1	General Troubleshooting . . . . .	80
10.2.2	Fusion and Workstation Troubleshooting . . . . .	81
10.2.3	Uninstallation . . . . .	83
<b>11</b>	<b>Appendix I: Command-Line Examples</b>	<b>83</b>
11.0.4	Getting help . . . . .	83
11.0.5	Interactive vs. Non-interactive mode . . . . .	84
11.0.6	IDs . . . . .	84
11.0.7	Setting a target . . . . .	84
11.0.8	Tenants . . . . .	85
11.0.9	Set tenant for other commands . . . . .	85
11.0.10	Resource tickets . . . . .	85
11.0.11	Projects . . . . .	86
11.0.12	Flavors . . . . .	86
11.0.13	Images . . . . .	87
11.0.14	VMs . . . . .	88
11.0.15	Hosts . . . . .	89
<b>12</b>	<b>Appendix II: Deployment Template</b>	<b>90</b>
12.0.16	Base Template . . . . .	90
12.0.17	Hosts on Disparate Networks . . . . .	91

12.0.18	Deployment with Authentication . . . . .	91
12.0.19	Host Metadata . . . . .	91
<b>13</b>	<b>Appendix III: Setting Up NSX Before Photon</b>	<b>93</b>
13.1	Prerequisites . . . . .	93
13.2	References . . . . .	93
13.3	Overview . . . . .	93
13.3.1	Fabric Topology . . . . .	93
13.3.2	Virtual Network Topology . . . . .	95
13.4	NSX Configuration . . . . .	97
13.4.1	Create VLAN Transport Zone . . . . .	97
13.4.2	Create Overlay Transport Zone . . . . .	99
13.4.3	Create Uplink Profile . . . . .	101
13.4.4	Create IP Address Pool . . . . .	103
13.4.5	Configure Edge Transport Node . . . . .	103
13.4.6	Create Edge Cluster . . . . .	105
13.4.7	Create Tier-0 Router . . . . .	105
13.5	DHCP Configuration . . . . .	111
13.5.1	Create DHCP Server Virtual Network . . . . .	111
13.5.2	Deploy DHCP Server . . . . .	119
13.6	Deploy Photon Controller . . . . .	119
<b>14</b>	<b>Appendix IV: Release Notes</b>	<b>120</b>
14.1	Photon Controller 1.1 Release Notes . . . . .	120
14.1.1	Supported Platforms . . . . .	120
14.1.2	Security . . . . .	120
14.1.3	vSAN Integration . . . . .	121
14.1.4	NSX Integration . . . . .	121
14.1.5	Container Orchestration Frameworks . . . . .	121
14.1.6	UI . . . . .	122
14.1.7	Photon Controller Agent . . . . .	122
14.1.8	Deployment . . . . .	122
14.1.9	API . . . . .	122
14.1.10	Other . . . . .	122

# 1 Photon Controller and Photon Platform

Photon Platform is a highly scalable multi-tenant control plane designed for cloud-native applications. The platform includes Photon Controller and ESXi. It provides an IaaS-style API to create, manage, and destroy virtual machines and container cluster frameworks like Kubernetes. Photon Platform is also available as a bundle with Pivotal Cloud Foundry. The design favors scale, high churn, and self-healing of the infrastructure.

Photon Controller is built for cloud-native applications. While capable of running other workloads, it is designed for modern workloads, including container-based applications.

## 1.1 Integration with VMware NSX

If you want to integrate Photon Controller with a key component of Photon Platform—VMware NSX—you must install NSX before you install Photon Controller. For more information, see the appendix on setting up NSX for Photon Controller.

## 2 Quick-Start Installation

### 2.1 Introduction

This guide explains how to install Photon Controller for demonstration or trial purposes. As a quick start guide, it focuses on a minimal configuration to set up Photon Controller 1.1 with authentication in a controlled environment.

Photon Controller forms part of VMware Photon Platform, a highly scalable multi-tenant control plane for cloud-native applications. Photon Platform includes VMware ESXi, Photon Controller, and Lightwave security services.

Photon Controller furnishes an API, a CLI, and a UI to manage infrastructure as a service. You can create virtual machines and Kubernetes clusters to securely run cloud-native applications and containerized workloads at scale.

#### 2.1.1 Overview

Photon Controller runs on ESXi and virtual machines in ESXi. You install Photon Controller by first downloading the Photon Controller installer—an OVA that creates a VM to which you connect to deploy the system’s infrastructure from a YAML configuration file.

The infrastructure of Photon Controller contains two main components:

- A management ESXi host composed of one or more virtual machines running on ESXi. The management plane allocates resources to tenants for projects.
- A cloud host that resides on an ESXi host to run your users’ VMs.

For expedience, this guide installs Photon Controller on an ESXi machine that holds the management plane and the cloud host.<sup>1</sup> After you install the system, you can use the management plane to create tenants, resource tickets, and projects.

To help you learn how to install Photon Controller, this quick start guide proceeds in two stages: The first stage installs a basic deployment of the system, and the second stage installs a more advanced deployment that uses Lightwave authentication.<sup>2</sup>

#### 2.1.2 Lightwave Authentication

Photon Controller integrates with [Lightwave](#) to help secure Photon Platform. An open source project published by VMware on GitHub, Lightwave furnishes a directory service, a certificate authority, a certificate store, and an authentication service. Lightwave authenticates users and groups with its directory service to ensure that only authorized users run authorized containers.

#### 2.1.3 Assumptions

This guide assumes that an ESXi host is in place with the following attributes.<sup>3</sup> The ESXi host can be either the licensed or the free version.

---

<sup>1</sup>For a production system, you would deploy the management plane as a cluster of 3 VMs.

<sup>2</sup>For security, you should deploy Photon Controller with authentication turned on.

<sup>3</sup>If you do not have a computer running the ESXi operating system, you can obtain ESXi at the [VMware vSphere Hypervisor 6.0 Download Center](#). For help installing it, see the instructions at the download center and in the [VMware vSphere 6 Documentation](#). You can also install Photon Controller on VMware Workstation or Fusion; for instructions, see the [Photon Controller GitHub Wiki](#).

- It is running VMware ESXi 6.0.0 Patch 201611001 (ESXi600-201611001), which you can find by searching for patches for ESXi 6.0.0 on the [My VMware Product Patches web site](https://my.vmware.com/group/vmware/patch) at <https://my.vmware.com/group/vmware/patch>.
- It contains no VMs and has at least 4 CPU cores and 8 GB of RAM.
- It is assigned a static IP address.
- It contains a datastore with read-write access.
- It is not managed with vCenter.

This guide also assumes that you have root access to the ESXi host, know how to manage it with the vSphere Web Client<sup>4</sup>, and understand basic networking in a virtualized environment. Only minimal instructions are provided for using ESXi and its web client; if you need help, see the VMware documentation for ESXi.

#### 2.1.4 Requirements

- Photon Controller version 1.1.
- A static IP address that you can use to provision the Photon Controller management node.
- A Linux workstation that can connect to virtual machines on your ESXi host to run commands with the Photon Controller command-line interface (CLI). This guide uses Ubuntu 14.04 as an example.<sup>5</sup>
- A second ESXi hypervisor with a static IP address that the installation will dedicate to the Lightwave authentication service. This second ESXi hypervisor can be embedded as a virtual machine in your primary ESXi host. You can download it for free from [VMware](#).

## 2.2 Installing the Photon Controller Installation OVA

Download the installer for Photon Controller 1.1—`installer-vm.ova`—from the following URL and then deploy it by using the vSphere Web Client:

<https://github.com/vmware/photon-controller/releases>

After you download it, quickly check its checksum to make sure the entire file downloaded correctly; the checksum resides at <https://github.com/vmware/photon-controller/wiki/Download>.

To deploy the OVA by using the vSphere Web Client, under **Navigator**, right-click **Host**, and then click **Create/Register VM**. Select **Deploy a virtual machine from an OVF or OVA file**, click **Next**, enter a name for the VM, such as `pc-installer`, and then click in the blue box to select the OVA from the directory that you downloaded it to.

Click **Next** and select a datastore; the examples in this guide assume that the datastore is named `datastore1`.

Move through the remaining dialog boxes by clicking **Next** to accept the default settings, and then click **Finish**.

When it finishes deploying, power on the virtual machine. In a later step, you will connect to it to create the management virtual machine and the cloud host in ESXi.

## 2.3 Installing the Photon Controller CLI on a Linux Workstation

Download the file named `photon-linux64` from the following URL and install it on a Linux workstation with which you can connect to your ESXi host.

<https://github.com/vmware/photon-controller/releases>

<sup>4</sup>The vSphere Web Client is also known as the VMware ESXi Host Client.

<sup>5</sup>You can also install the Photon Controller CLI on a Microsoft Windows or Mac workstation; for instructions, see the [Photon Controller GitHub Wiki](#).

To install it on Ubuntu, for example, change its mode bits so that it is executable and then move it to `/usr/local/bin/photon`. Here's an example:

```
cd ~/Downloads/  
chmod +x photon-linux64  
sudo mv photon-linux64 /usr/local/bin/photon
```

In a later step, you connect from the Linux workstation to the installer VM through the Photon CLI tool to run installation commands.

## 2.4 Preparing ESXi for Photon Controller

You must prepare ESXi for Photon Controller before you can proceed with the installation.

- Make sure that the default VLAN in ESXi named **VM Network** has available to it at least 1 unused IP address that you can assign as a static IP address. The IP address is for the VM that will act as the Photon Controller management VM.
- Make sure that the pool of IP addresses available to **VM Network** is large enough to support provisioning several VMs later.
- Set up an NTP source. For instructions, see the VMware vSphere documentation.
- Make sure that there are no more than two DNS entries; if there are three DNS entries, delete one of them by running the following commands. First, list the DNS entries: `esxcli network ip dns server list`. Second, remove one of them: `esxcli network ip dns server remove --server <IP-address>`.
- Turn on SSH: Connect to the ESXi host by using the vSphere Web Client. Under **Navigator**, right-click **Host**, click **Services**, and then click **Enable Secure Shell (SSH)**.

## 2.5 Preparing a YAML Configuration File

The crux of the installation revolves around understanding the YAML installation template and configuring it with the right values for your deployment.

The YAML file is, in effect, a manifest for the installation: The installer VM uses the values in the YAML file to connect to the ESXi host, identify the correct networking settings, locate the datastore, and set up the management node and cloud host.

Understanding each field in the YAML template will help expedite the installation. This section describes the YAML template and provides an example of a completed YAML configuration file.

### 2.5.1 The Anatomy of a Configuration File

Here is a minimal template with descriptions of each key-value pair. The template contains two main sections: **hosts** and **deployment**. The **hosts** section specifies networking information for all the computers, or “hosts,” in the cluster, including the ESXi hypervisor host and the virtual machines forming the management plane. The **deployment** section specifies parameters for the setup.

**hosts:**

```
- metadata:  
  MANAGEMENT_DATASTORE: <the name of a datastore on an  
    ESXi hypervisor that the management plane will use>  
  MANAGEMENT_PORTGROUP: <the name of a network (VLAN or  
    port group) on ESXi where the management VMs will reside>  
  MANAGEMENT_NETWORK_NETMASK: <the netmask of the  
    network on ESXi where the management VMs reside>
```



```

MANAGEMENT_NETWORK_DNS_SERVER: <the IP address of the
  DNS server of the network on ESXi to be used by the management VMs>
MANAGEMENT_NETWORK_GATEWAY: <the IP address of the
  gateway of the network on ESXi to be used by the management VMs>
MANAGEMENT_VM_IPS: <the static IP address for the
  management VM that the installer will create>
address_ranges: <the IP address of the ESXi host on
  which the management plane will be installed>
username: <the name of a user with root access to the ESXi host>
password: <the password for the user on the ESXi host>
usage_tags:
  - MGMT
  - CLOUD
deployment:
  resume_system: true
  image_datastores:
    - datastore1
  use_image_datastore_for_vms: true
  loadbalancer_enabled: false
  auth_enabled: false

```

In this minimal deployment template, the `CLOUD` usage tag indicates that the ESXi host will also serve as the cloud host. As a result, there is no need to add an additional `metadata` sequence for the cloud host.

## 2.5.2 Avoiding Errors in the Configuration File

A few of the fields in the template are error prone.

The value of the `MANAGEMENT_VM_IPS` key refers to the static IP address of the VM that the Photon Controller installer will deploy for the management node. The value that you add for this key should be an unused static IP address that's available to your ESXi host and reachable from the installer VM. You might need to obtain a static IP address from your network administrator.

The value of the `address_ranges` key refers to the IP address of your ESXi host on which you are installing Photon Controller. If you install Photon Controller across a cloud of ESXi hosts, the value of `address_ranges` can be the range of static IP addresses assigned to the ESXi hosts.

The value of `MANAGEMENT_PORTGROUP` is error prone because there are several terms that, in the context of ESXi networking as viewed through the vSphere Web Client, can seemingly refer to the same thing: *network*, *port group*, and *VLAN*. The default port group on an ESXi hypervisor is named `VM Network`, which often has a VLAN ID of 0, and `VM Network` is a suitable value for `MANAGEMENT_PORTGROUP` for a trial installation.

Port groups	Virtual switches	Physical NICs	VMkernel NICs	TCP/IP stacks	Firewall rules
Add port group          Edit settings                     Refresh                     Actions <span style="float: right;">Q Search</span>					
Name	Active ...	VLAN ID	Type	vSwitch	VMs
VM Network	1	0	Standard port group	vSwitch0	5
Management Network	1	0	Standard port group	vSwitch0	N/A
					2 items

Figure 1: Port Groups in the vSphere Web Client

In this context, the term *Management Network* can refer to two entirely different networks:

1. The default ESXi management network containing the Ethernet network adapter that transmits traffic between the ESXi host and any external management software. This management network is the one shown in the image above. In the YAML template, “MANAGEMENT\_NETWORK” does not refer to the ESXi Management Network.
2. The Photon Controller management plane, or “network,” that you are setting up.

In the YAML template, therefore, the values for the `MANAGEMENT_NETWORK_NETMASK`, the `MANAGEMENT_NETWORK_DNS_SERVER`, and the `MANAGEMENT_NETWORK_GATEWAY` all refer to the network infrastructure that you want Photon Controller to use. To find the networking information for this infrastructure, click **Host** in the navigation pane of the vSphere Web Client. It displays a summary of its settings and IP addresses, as the following image illustrates.

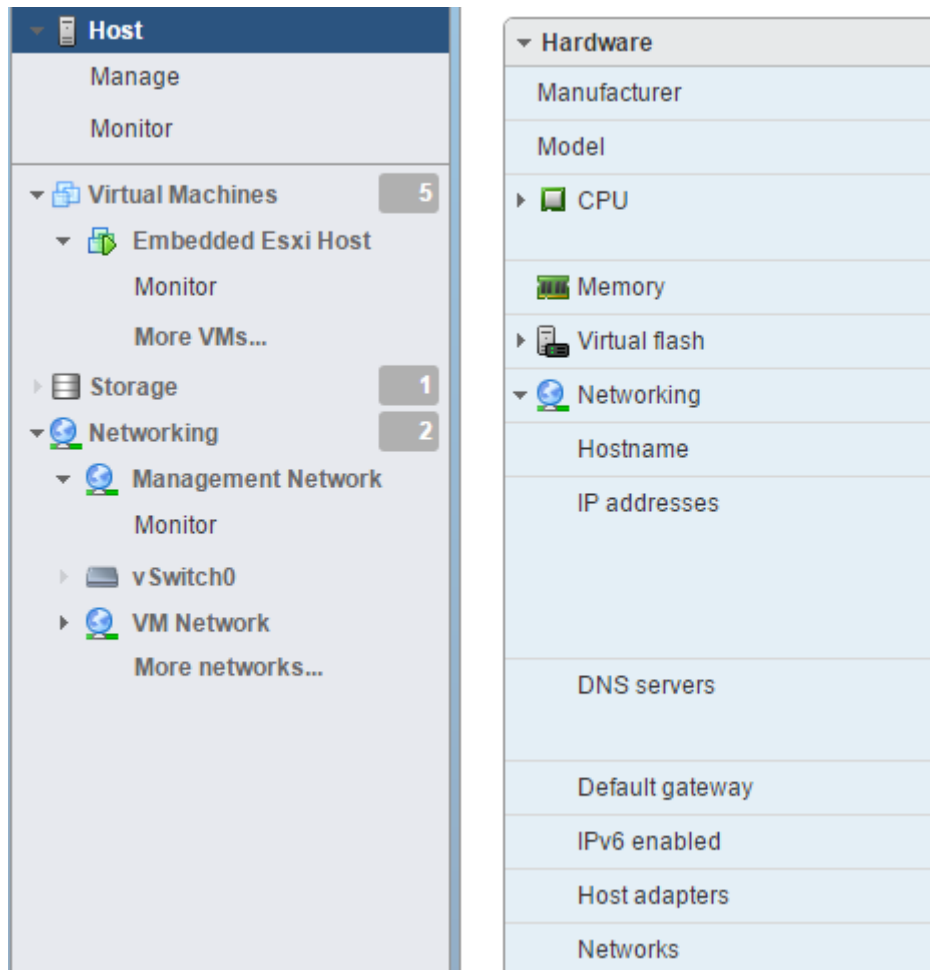


Figure 2: The ESXi Host information page in the vSphere Web Client

### 2.5.3 Example Basic Configuration File

Here is an example configuration file with the minimum information necessary for a basic deployment:

```
hosts:
  - metadata:
      MANAGEMENT_DATASTORE: datastore1
      MANAGEMENT_PORTGROUP: VM Network
      MANAGEMENT_NETWORK_NETMASK: 255.255.0.0
      MANAGEMENT_NETWORK_DNS_SERVER: 198.51.100.1
      MANAGEMENT_NETWORK_GATEWAY: 198.51.100.253
      MANAGEMENT_VM_IPS: 198.51.100.191
      address_ranges: 198.51.100.44
      username: root
      password: secret$1
      usage_tags:
        - CLOUD
        - MGMT
  deployment:
      resume_system: true
      image_datastores:
        - datastore1
      use_image_datastore_for_vms: true
      loadbalancer_enabled: false
      auth_enabled: false
```

For your first test deployment, copy the template above and replace the IP addresses and the username and password with those from your ESXi host and its network. The IP address for the value of `MANAGEMENT_VM_IPS` should be a static IP address that is available for use by the ESXi host. Save the file as `config1.yml`.

You can obtain the information to fill out the template by connecting to your ESXi host with the vSphere Web Client.

## 2.6 Deploying the Basic System

You can now use the configuration file that you created to deploy a basic system from the Photon Controller CLI on your Linux workstation.

First, set the target for the Photon CLI tool by running the following command, replacing `<installer_vm_ip>` with the IP address of the Photon installation VM that you deployed earlier from the OVA. Add Port 9000 after the IP address.

```
photon target set http://<installer_vm_ip>:9000
```

You are now ready to deploy the Photon Controller system from your Linux workstation by running the following command:

```
photon system deploy config1.yml
```

If the deployment was unsuccessful, see the section on troubleshooting.

## 2.7 Checking the System's Status

First, change the target of the CLI tool to the static IP address of the management VM:

```
photon target set http://<management_vm_static_ip_address>:9000
```

Example:

```
photon target set http://198.51.100.44:9000
```

Second, check the system's status; example:

```
photon system status
Overall status: READY
Component      Status
PHOTON_CONTROLLER  READY
```

Finally, connect to the Photon Controller user interface by starting a web browser on your Linux workstation and going to the IP address of the management VM:

```
http://<mgt_ip>
```

Or, you can browse the API:

```
http://<mgmt_ip>:9000/api
```

When you're done checking the system out, you should delete it because authentication is disabled. To remove the system, run the following command in the Photon CLI on your Linux workstation:

```
photon system destroy
```

The next section demonstrates how to redeploy Photon Controller as a secured system.

## 2.8 Redeploying with Lightwave Authentication

This section shows you how to tear down the previous deployment of Photon Controller and redeploy it with Lightwave authentication. The redeployment builds on the knowledge you acquired during the first deployment.

Setting up Photon Controller with authentication requires you to add items to the YAML configuration file and to add a nested ESXi virtual machine to the ESXi host. You will also need a static IP address that you can dedicate to the VM running Lightwave.

### 2.8.1 Destroying the Previous Deployment

Before adding authentication, make sure you permanently erase the Photon Controller system that you deployed in the previous sections by running the following command in the Photon CLI on your Linux workstation:

```
photon system destroy
```

### 2.8.2 Preparing ESXi for Authentication

Deploying Photon Controller with authentication requires a second ESXi hypervisor dedicated to providing security services. For the purposes of installing Photon Controller with authentication quickly, this guide embeds a second ESXi host as a virtual machine in the primary ESXi host.

To support running the Lightwave authentication service on the nested ESXi host, you must enable **promiscuous mode** on the primary, physical ESXi host: In the **Navigator** pane of the vSphere Web Client, click **Networking**, click the vSwitch for VM Network (typically, vSwitch0), click **Actions**, click **Edit Settings**, expand **Security**, and then, for **Promiscuous mode**, select **Accept**.

The nested ESXi host has several requirements:

- 5 GB of RAM

- 150 GB of disk space
- A datastore with a different name than that of the primary ESXi host; in this example installation, the name will be `datastore2`. The datastore names must be unique so that each name maps to one actual datastore.
- A change to the VM's CPU settings to enable ESXi virtualization.
- SSH is enabled.

This section demonstrates how to fulfill these requirements as you install an ESXi host as a VM embedded in the primary ESXi host.

First, go to the [VMware vSphere Hypervisor 6.0 Download Center](#) and download the free version of ESXi 6.0 Update 2. Make sure it includes VMware ESXi 6.0.0 Patch 201611001 (ESXi600-201611001), which is available by searching for patches for ESXi 6.0.0 on the [My VMware Product Patches web site](https://my.vmware.com/group/vmware/patch) at <https://my.vmware.com/group/vmware/patch>.

Second, [upload the ISO](#) for ESXi to the datastore of your primary ESXi host.<sup>6</sup>

Now deploy a VM for the nested ESXi host by connecting to the primary ESXi host with the vSphere Web Client:

First, under **Navigator**, right-click **Host**, and then click **Create/Register VM**. On the **Select a creation type** page, select **Create a new virtual machine** and click **Next**.

Second, on the **Select a name and guest OS** page, enter a name for the VM, such as **Embedded ESXi Host for Auth**. For the **Guest OS family**, select **Linux**. For the **Guest OS version**, select **Other Linux (64-bit)**. Click **Next**.

Third, on the **Select storage** page, leave the datastore set to its default setting; click **Next**.

Fourth, on the **Customize settings** page, expand the **CPU** section and select **Expose hardware assisted virtualization to the guest OS**. Expand the **Memory** section and change the RAM to 5 GB. Next, change the setting for **Hard disk 1** to 150 GB. For the **CD/DVD Drive 1** setting, select **Datastore ISO file**, browse to the ESXi ISO file that you uploaded earlier, and then click **Select**. The ESXi ISO file name looks similar to this:

```
VMware-VMvisor-Installer-6.0.0-4558704.x86_64.iso
```

Click **Next**, and then click **Finish**.

When it finishes deploying, power on the virtual machine, enable SSH on this new nested ESXi host, and then change the name of its datastore to `datastore2`.

### 2.8.3 Adding Lightwave Authentication to the Template

In the example above, the value for authentication is set to `false`. Turning on authentication requires the following additions to the YAML configuration file:

- A second `metadata` subsection that describes the networking information and credentials for the Lightwave authentication service.
- A static IP address for the nested ESXi host on which the authentication service will run.
- The name of the datastore on Lightwave's ESXi host; the name of its datastore must be different from the name of the datastore used by the management VM.
- Additional entries in the `deployment` section for authentication.

The new `metadata` section adds a key for `ALLOWED_SERVICES` set to `Lightwave`. The management VMs in this `metadata` sequence are dedicated to the Lightwave authentication service, and their networking information must be different from those in the metadata sequence for the management plane. Here is an example of a `metadata` sequence for authentication:

---

<sup>6</sup>For instructions, see [Upload Files to Datastores](#).

```
- metadata:
  ALLOWED_SERVICES: Lightwave
  MANAGEMENT_DATASTORE: datastore2
  MANAGEMENT_PORTGROUP: VM Network
  MANAGEMENT_NETWORK_NETMASK: 255.255.0.0
  MANAGEMENT_NETWORK_DNS_SERVER: 198.51.100.1
  MANAGEMENT_NETWORK_GATEWAY: 198.51.100.253
  MANAGEMENT_VM_IPS: 198.51.100.244
  address_ranges: 198.51.100.134
  username: root
  password: secret$1
  usage_tags:
    - MGMT
    - CLOUD
```

You also must add items to the `deployment` section for authentication:

```
deployment:
  resume_system: true
  image_datastores:
    - datastore1
    - datastore2
  auth_enabled: true
  oauth_tenant: 'esxcloud'
  oauth_username: 'Administrator'
  oauth_password: 'LightWave!'
  oauth_security_groups:
    - "esxcloud\Administrators"
    - "esxcloud\photonControllerAdmins"
  sdn_enabled: false
  stats_enabled: false
  use_image_datastore_for_vms: true
  loadbalancer_enabled: true
  ntp_endpoint: 203.0.113.1
```

Here's what the relevant key-value pairs mean:

**resume\_system:** This setting determines whether Photon Controller should continue operating after deployment; set it to `true`.

**auth\_enabled:** Setting the value of this key to `true` turns on authentication.

**oauth\_tenant:** The Photon Controller security domain; it must be set to `esxcloud`. All letters must be lowercase.

**oauth\_username:** The user name of the Lightwave administrator.

**oauth\_password:** The password for the Lightwave administrator. Change the value to a strong password that you will remember.

**oauth\_security\_groups:** A list of groups whose members receive system administrator rights on Photon Controller.

## 2.8.4 Example Configuration File with Authentication

Here is an example configuration file that includes key-value pairs and sequence items for authentication.

`hosts:`

```

- metadata:
  MANAGEMENT_DATASTORE: datastore1
  MANAGEMENT_PORTGROUP: VM Network
  MANAGEMENT_NETWORK_NETMASK: 255.255.0.0
  MANAGEMENT_NETWORK_DNS_SERVER: 198.51.100.1
  MANAGEMENT_NETWORK_GATEWAY: 198.51.100.253
  MANAGEMENT_VM_IPS: 198.51.100.46
  address_ranges: 198.51.100.44
  username: root
  password: Secret$1
  usage_tags:
    - MGMT
    - CLOUD
- metadata:
  ALLOWED_SERVICES: Lightwave
  MANAGEMENT_DATASTORE: datastore2
  MANAGEMENT_PORTGROUP: VM Network
  MANAGEMENT_NETWORK_NETMASK: 255.255.0.0
  MANAGEMENT_NETWORK_DNS_SERVER: 198.51.100.1
  MANAGEMENT_NETWORK_GATEWAY: 198.51.100.253
  MANAGEMENT_VM_IPS: 198.51.100.244
  address_ranges: 198.51.100.134
  username: root
  password: secret1
  usage_tags:
    - MGMT
    - CLOUD
deployment:
  resume_system: true
  image_datastores:
    - datastore1
    - datastore2
  auth_enabled: true
  oauth_tenant: 'esxcloud'
  oauth_username: 'Administrator'
  oauth_password: 'LightWave!'
  oauth_security_groups:
    - "esxcloud\Administrators"
    - "esxcloud\photonControllerAdmins"
  sdn_enabled: false
  stats_enabled: false
  use_image_datastore_for_vms: true
  loadbalancer_enabled: true
  ntp_endpoint: 203.0.113.1

```

## 2.8.5 Building Your Configuration File and Deploying Photon Controller

Armed with these examples of how to add authentication to the YAML template, you can now rename your original deployment file `config2.yml` and modify it to include authentication. Or you can use the example configuration in the previous section as a template for your deployment. Replace the IP addresses, usernames, passwords, and datastore names in the template with those from your environment.

After you complete your YAML file, deploy it from your Linux workstation:

```
photon target set http://<installer_VM_ip>:9000
photon system deploy config2.yml
```

If the deployment was unsuccessful, see the section on troubleshooting.

## 2.9 Creating Accounts in Lightwave

After you deploy Photon Controller with authentication, you must log in to the Lightwave service and create a Photon Controller system administrators group that contains a user. When you're done, the authenticated user can create [tenants](#), [projects](#), and [users](#) on Photon Controller.

First, find the end point IP address of the Lightwave container by running the following commands with the Photon CLI on your Linux workstation:

```
photon deployment show
```

Connect to the Lightwave VM with SSH; the default password is `vmware`:

```
ssh esxcloud@<IPAddressOfAuthEndPoint>
```

Connect to the Lightwave container:

```
docker exec -it Lightwave bash
```

Change directories:

```
cd /opt/vmware/bin
```

Using the password defined in the `oauth_password` key of your deployment template (`Lightwave!` in the example), run the following Lightwave directory commands to create a group, create a user, and add the user to the group.<sup>7</sup>

```
./dir-cli ssogroup create --name "photonControllerAdmins"
./dir-cli user create --account pc-admin --user-password 'Your$ecret1!'
                        --first-name pc --last-name admin
./dir-cli group modify --name photonControllerAdmins --add pc-admin
```

Type `exit` to leave the container, and then type `exit` again to leave the SSH session.

## 2.10 Connecting to the Load Balancer

After the installer deploys Photon Controller and you created a security group in Lightwave, you can connect to the load balancer to create tenants, resource tickets, and projects.

First, find the load balancer's IP address by running the following commands with the Photon CLI on your Linux workstation:

```
photon deployment show
```

Since you deployed Photon Controller with authentication, you must connect to the IP address of the load balancer by appending Port 443:

```
photon target set -c https://<production_system_ip>:443
```

And then you can log in by using the `pc-admin` account at `oauth_tenant` that you created in the Lightwave directory. Lightwave authenticates the user.

```
photon target login --username pc-admin@<oauth_tenant> --password 'Your$ecret1!'
```

---

<sup>7</sup>The second command in the series should be run as one line; here it wraps onto a second line to fit the page.



Here is an example. In the sample YAML file, the `oauth_tenant` is set to `esxcloud`—so `<oauth_tenant>` is replaced with `esxcloud`:<sup>8</sup>

```
photon target login --username pc-admin@esxcloud --password 'Your$ecret1!'
```

Finally, check your work:

```
photon system status
Overall status: READY
Component      Status
PHOTON_CONTROLLER  READY
```

As the status says, you're now ready to work with the system.

## 2.11 Creating a Kubernetes Cluster

To see the power of Photon Controller, you can create a Kubernetes cluster. For instructions, see [Creating a Kubernetes Cluster](#) on the [Photon Controller GitHub wiki](#).

## 2.12 Troubleshooting

If Photon Controller fails to install successfully, clean up the unsuccessful installation by destroying it before you try again:

```
photon system destroy
```

You can then troubleshoot by looking at the installer's logs. The most likely cause of a failure is that IP addresses assigned to a VM in the Photon Controller management plane are in use, unavailable, or unreachable.

### 2.12.1 Log Files for the Installer

The password for the root account on the Photon Controller installer VM is `vmware`. The log files for the deployment reside in the following directory on the installer VM:

- `/var/log/esxcloud/photon-controller-core/`

The log files for the Photon Controller agent, which is installed on the ESXi host, reside in the following directory of the target ESXi host:

- `/scratch/log/photon-controller-agent.log`

### 2.12.2 Deploying the OVA Installer

If you get a failure screen when you deploy the OVA by using the vSphere Web Client, download and install the latest VIB for the web client from this URL:

<https://labs.vmware.com/flings/esxi-embedded-host-client>

---

<sup>8</sup>If you set `oauth-tenant` to something other than `esxcloud`, make sure all the letters are lowercase.

### 2.12.3 Lightwave Authentication and NTP

Because Lightwave authenticates users and groups by using the Kerberos security protocol, the time on the VM running Lightwave must be synchronized with the time on VMs that are authenticating with Lightwave.

More specifically, the clock of the client must be within the Lightwave key distribution center's maximum clock skew, which is 300 seconds, or 5 minutes, by default. Lightwave discards authentication requests outside the maximum clock skew to help prevent replay attacks. See [MIT Kerberos Clock Skew](#).

Implementing an NTP server for the ESXi host synchronizes clocks across virtual machines to avoid Kerberos clock-skew errors.

## 3 Authentication and Authorization

### 3.1 Authenticating Multitenant Users and Groups

Here's how to work with Lightwave and Photon Controller to authenticate and authorize users and groups in the context of Photon's model of multitenancy.

#### Table of Contents

- [Lightwave Security Services](#)
- [Roles and Rights](#)
- [Process Overview](#)
- [Connecting to the Lightwave Management VM](#)
- [Gaining Access to the Docker Container Running Lightwave](#)
- [Creating a System Administrator in Lightwave](#)
- [Creating a Tenant Administrator](#)
- [Creating Project Users](#)
- [Using Lightwave Groups in Photon Controller](#)
- [Setting Security Groups for Administrators, Tenants, and Project Users](#)
- [Connecting to the Load Balancer for Secure Login](#)
- [Tenants, Resource Tickets, and Projects](#)
- [Projects](#)

#### 3.1.1 Lightwave Security Services

Photon Controller integrates with [Lightwave](#) to help secure Photon Platform. An open source project published by VMware on GitHub, Lightwave furnishes a directory service, a certificate authority, a certificate store, and an authentication service.

Lightwave authenticates Photon Platform users with its directory service. Lightwave group memberships provide a means of authorizing Photon Platform users as system administrators, tenants, or project users.

#### 3.1.2 Roles and Rights

Photon Controller's multitenant model includes the following types of users:

**System administrators.** The Photon Controller system administrator has rights to perform any action. They create tenants and establish the security group for each tenant.

**Tenant administrators.** A system administrator assigns at least one tenant administrator to each tenant. Under the tenant to which they are assigned, a tenant administrator can create, modify, and delete resource

tickets, projects, and other objects scoped under a tenant or a project. A tenant administrator can also manage the security groups associated with the tenant.

**Project users.** Project users can view and modify project resources, including Kubernetes clusters, VMs, and disks. After a Photon Controller tenant administrator or system administrator sets a security group for a project, the group members are granted project user rights.

### 3.1.3 Process Overview

The process of creating users and groups goes like this:

1. Find the virtual machine running Lightwave so that you can access Lightwave to create users and groups.
2. Connect to the Docker container running Lightwave to use the directory command-line utility.
3. Create a system administrator group and user for Photon Controller so that you can log on to Photon Controller and be authenticated with Lightwave.
4. Optionally create additional users and groups in Lightwave by using the Lightwave directory utility.
5. Log in to Photon Controller as the system administrator, a tenant administrator, or a project user.

### 3.1.4 Connecting to the Lightwave Management VM

Before you can create users and groups in Lightwave for use in Photon Controller, you must find the Lightwave management VM, connect to it with SSH, and then enter the Docker container running the Lightwave service.

First, find the end point IP address of the Lightwave container by running the following commands with the Photon command-line utility on a workstation that's connected to Photon Controller:

```
photon deployment show
```

Connect to the Lightwave VM with SSH; the default password is `vmware`.

```
ssh esxcloud@<IPAddressOfAuthEndPoint>
```

### 3.1.5 Gaining Access to the Docker Container Running Lightwave

After logging in to the Lightwave VM with SSH, you can connect to the Docker container running the Lightwave service:

```
docker exec -it Lightwave bash
```

Change directories to the directory containing the Lightwave directory management command-line utility, called `dir-cli`:

```
cd /opt/vmware/bin
```

You can now use the Lightwave `dir-cli` tool to create a system administrator, a tenant administrator, and a project user for Photon Controller by using the following sequence of commands:

```
./dir-cli ssogroup create --name "NameOfGroup"
./dir-cli user create --account <account-name> --user-password 'Your$ecret1!'
                        --first-name <firstName> --last-name <lastName>
./dir-cli group modify --name <NameOfGroup> --add firstName-lastName
```

### 3.1.6 Creating a System Administrator in Lightwave

After you first deploy Photon Controller with authentication turned on, you must log in to the Lightwave management VM and create a Photon Controller system administrators group that contains a user. When you're done, the user is a system administrator who can create tenants and project users on Photon Controller.

Using the password defined in the `oauth_password` key of your deployment template (**Lightwave!** in the example in the [quick start guide](#)), you can run Lightwave directory commands similar to the following examples to create a group in Lightwave, create a user, and add the user to the group.

```
./dir-cli ssogroup create --name "photonControllerAdmins"
./dir-cli user create --account pc-admin --user-password 'Your$ecret1!'
                        --first-name pc --last-name admin
./dir-cli group modify --name photonControllerAdmins --add pc-admin
```

### 3.1.7 Creating a Tenant Administrator

Similarly, you can use the same sequence of commands to create a tenant administrators group and a tenant administrator:

```
./dir-cli ssogroup create --name tenant-admins
./dir-cli user create
    --account demo-tenant-admin
    --user-password Passw0rd!
    --first-name demo
    --last-name tenant
./dir-cli group modify --name tenant-admins --add demo-tenant-admin
```

### 3.1.8 Creating Project Users

And here's an example of how to create a group and a user for a project in Photon Controller:

```
./dir-cli ssogroup create --name dev-project-users
./dir-cli user create
    --account dev-project-user1
    --user-password Passw0rd!
    --first-name Project
    --last-name User1
./dir-cli group modify --name dev-project-users --add dev-project-user1'
```

When you're done creating groups and users, type **exit** to leave the container, and then type **exit** again to leave the SSH session.

For more information about Lightwave, see the [Lightwave GitHub repo](#).

### 3.1.9 Using Lightwave Groups in Photon Controller

Photon Controller uses Lightwave groups as generic collections of user accounts. The groups are created by a Lightwave administrator, not the Photon Controller administrator. In Lightwave, the groups contain no properties that distinguish system administrators from tenant administrators or project users.

To control access by enforcing the distinctions among groups in Photon Controller, you use one of the following commands to set a group from Lightwave as the security group for a deployment, tenant, or project.

```
photon deployment set-security-groups
```

Members of the security group for a deployment are system administrators for Photon Controller.

```
photon tenant set_security_groups
```

Members of the security groups for a tenant are tenant administrators.

```
photon project set_security_groups
```

Members of the security groups for a project are project users. They receive project-specific rights to work with and modify projects.

**Important:** The command to set security groups *overrides* existing groups. The groups that you include in the `photon project set_security_groups` command, for example, replace all the existing security groups—even ones that you have already defined. The only exception is inherited groups, which are retained by default. When you create a project, the project inherits the security groups from the tenant that governs the project, including whatever groups the tenant inherited.

The next section presents examples of how to set the security groups for a deployment, a tenant, and a project. Remember to be careful when you run the commands to set a security group so that you don't lock yourself out of the system.

### 3.1.10 Setting Security Groups for Administrators, Tenants, and Project Users

To set a group named `dev-project-users` as the security group for an existing project named `dev-project`, connect to Photon Controller by using the Photon command-line utility on your workstation and then run the following command:

```
photon project set_security_groups dev-project -t demo -g photon\dev-project-users
```

As the example above illustrates, you specify group names in the following format: `<securitydomain>\<NameOfGroup>`. Here's an example: `photon\Administrators`. The security domain must be made up of all lowercase letters.

Here's an example of how to set a group as a tenant administrator. All the members of the group have tenant administrator rights. In the example, `plato` is the name of the tenant.

```
photon tenant set_security_groups plato photon\tenant-admins
```

A system administrator can set a Lightwave group that contains a list of users who have rights to administer the entire deployment of Photon Controller. **Important:** The following command overrides the existing groups. Be careful running it because providing the wrong groups could remove your access:

```
photon deployment set-security-groups photon\photonControllerAdmins
```

All the commands to set security groups can contain a comma-separated list of groups; example:

```
photon deployment set-security-groups photon\photonControllerAdmins,  
    photon\Administrators, photon\superUsers
```

### 3.1.11 Connecting to the Load Balancer for Secure Login

After you created a security group in Lightwave, you can connect to the load balancer to create tenants, resource tickets, and projects.

First, find the load balancer's IP address by running the following commands with the Photon CLI on your workstation:

```
photon deployment show
```

Since you deployed Photon Controller with authentication, you must connect to the IP address of the load balancer by appending Port 443:

```
photon target set -c https://<production_system_ip>:443
```

And then you can log in by using the `pc-admin` account at `oauth_tenant` that you created in the Lightwave directory. Lightwave authenticates the user.

```
photon target login --username pc-admin<oauth_tenant> --password 'Your$ecret1!'
```

Here is an example. In the sample YAML file used by the quick start guide, the `oauth_tenant` is set to `esxcloud`—so `<oauth_tenant>` is replaced with `esxcloud`. (You may have used a different name for the `oauth_tenant`, such as `photon`.)

```
photon target login --username pc-admin@esxcloud --password 'Your$ecret1!'
```

Finally, check your work:

```
photon system status
Overall status: READY
Component      Status
PHOTON_CONTROLLER  READY
```

As the status says, you're now ready to work with the system. You can create tenants, projects, and other resources.

### 3.1.12 Tenants, Resource Tickets, and Projects

As a multitenant system, Photon Controller abstracts physical resources so that it can allocate them across the system on behalf of users. In Photon Controller's multitenant structure, tenants, resource tickets, and project users are closely tied together.

When you create a tenant, you give it a pool of resources that the tenant's projects can consume. The pool is allocated through resource tickets. A resource ticket sets aside a limited amount of resources, such as CPUs, RAM, and disks.

The limits for each class of resource are defined as a tuple that includes a key, a value, and a unit.

Key	Value (type)	Unit	Description
vm	integer	COUNT	The number of Virtual Machines that can be created by all projects using this ticket
vm.memory	integer	GB	The amount of virtual memory, in gigabytes, that can be consumed by all virtual machines in all projects using this ticket
vm.cpu	integer	COUNT	The number of virtual CPU cores that can be consumed by all virtual machines in all projects using this ticket

A resource ticket is scoped to a single tenant, persists for the life of the tenant, and is removed when the tenant is deleted.

### 3.1.13 Projects

A project is a way to allocate the pool of resources assigned to a tenant through resource tickets.

Tenant administrators create projects based on a resource ticket and can impose project-specific limits on ticketed resources. A project can be granted the right to use all or part of a ticket's resources. Project users can create VMs and disks by paring images with flavors.

For more information about tenants, projects, and resources, see [Working with Tenants, Resource Tickets, and Projects](#).

## 3.2 The Authorization Model

The authorization model separates system administrators, tenant administrators, and project users.

System administrators are permitted to read and modify all the objects in the system. They can also create tenant administrators. A tenant administrator can view and modify projects, resource tickets, and other objects owned by the tenant. A tenant administrator or a system administrator can create project users, who can view and modify project resources, such as VMs, disks, and images.

### 3.2.1 System Administrator

These users have full access to all APIs in the system.

The table below describes APIs that are to be used by System Administrators and their attributes:

#### 3.2.1.1 API Calls by System Administrators

Only a system administrator can access these API endpoints.

API Call	Description
<b>/datastores</b>	List datastores in use by Photon Controller
<b>/deployments</b>	List Photon Controller deployments
<b>/hosts</b>	List hosts under control of Photon Controller
<b>/portgroups</b>	List portgroups available
<b>/status</b>	List Photon Controller's system status

#### 3.2.1.2 Sysadmin R/W; Others R/O - API Calls

Only a system administrator can write to these API endpoints; all other users may read them. By “write,” we mean create, modify, and delete—“manage” in the table below.

API Call	Sysadmin Actions	Actions for others
<b>/flavors</b>	Manage flavors	List/show flavors
<b>/networks</b>	Manage networks	List/show networks
<b>/tenants</b>	Manage tenants	List/show tenants
<b>/resource-tickets</b>	Manage tickets	List/show tickets

### 3.2.2 Tenant Administrator

Tenant Administrators are assigned on a per-tenant basis and have the following capabilities:

- Creating and deleting projects under the tenant they are assigned
- Creating and deleting resource tickets
- Managing the security groups associated with the tenant
- Fully manipulating any object scoped under the tenant and project

#### 3.2.2.1 API Calls by Tenant Administrator

A tenant administrator can manage the the projects within the tenant.

API Call	Description
<code>/tenants/MY-TENANT/set_security_groups</code>	Manage security groups for MY-TENANT
<code>/tenants/MY-TENANT/projects</code>	Manage projects for MY-TENANT

### 3.2.3 Project Users

Project users can view and modify project resources, including VMs and disks. After a Photon Controller tenant administrator or system administrator binds a security group to a project, all members of that group are granted project user rights.

#### 3.2.3.1 API Calls by Project Users

API Call	Description
<code>/projects/MY-PROJECT/clusters</code>	Manage container clusters for MY-PROJECT
<code>/projects/MY-PROJECT/disks</code>	Manage disks for MY-PROJECT
<code>/projects/MY-PROJECT/vms</code>	Manage VMs for MY-PROJECT
<code>/projects/MY-PROJECT/set_security_groups</code>	Manage security groups for MY-PROJECT

For more information about tenants and resources, see [Understanding Multitenancy](#) and [Working with Tenants, Resource Tickets, and Projects](#).

## 4 Basic Operations

### 4.1 Connecting the Photon Load Balancer or Management VM

You can log in to Photon Controller by connecting to the load balancer. First, find the load balancer's IP address by running the following commands with the Photon command-line utility on your workstation:

```
photon deployment show
```



If you deployed Photon Controller with authentication, you must connect to the IP address of the load balancer by appending Port 443:

```
photon target set https://<production_system_ip>:443
```

If you deployed Photon Controller without authentication turned on, connect to Port 9000.

```
photon target set http://<production_system_ip>:9000
```

For security, VMware recommends that you run Photon Controller only with authentication turned on.

## 4.2 Logging In to Photon Controller

And then you can log in by using an account that you created in the Lightwave directory. Lightwave authenticates the user. Here's an example with a user named pc-admin:

```
photon target login --username pc-admin@esxcloud --password 'Your$ecret1!'
```

After you log in, check the system's status:

```
photon system status
Overall status: READY
Component      Status
PHOTON_CONTROLLER  READY
```

As the status says, you're now ready to work with the system by creating tenants, resource tickets, and projects.

## 4.3 Connecting to the Web UI

You can also log into the web interface to view a range of information:

```
http://<ip-address-of-load-balancer>
```

Here's what the web interface looks like after Photon Controller was installed but no resources have been created:

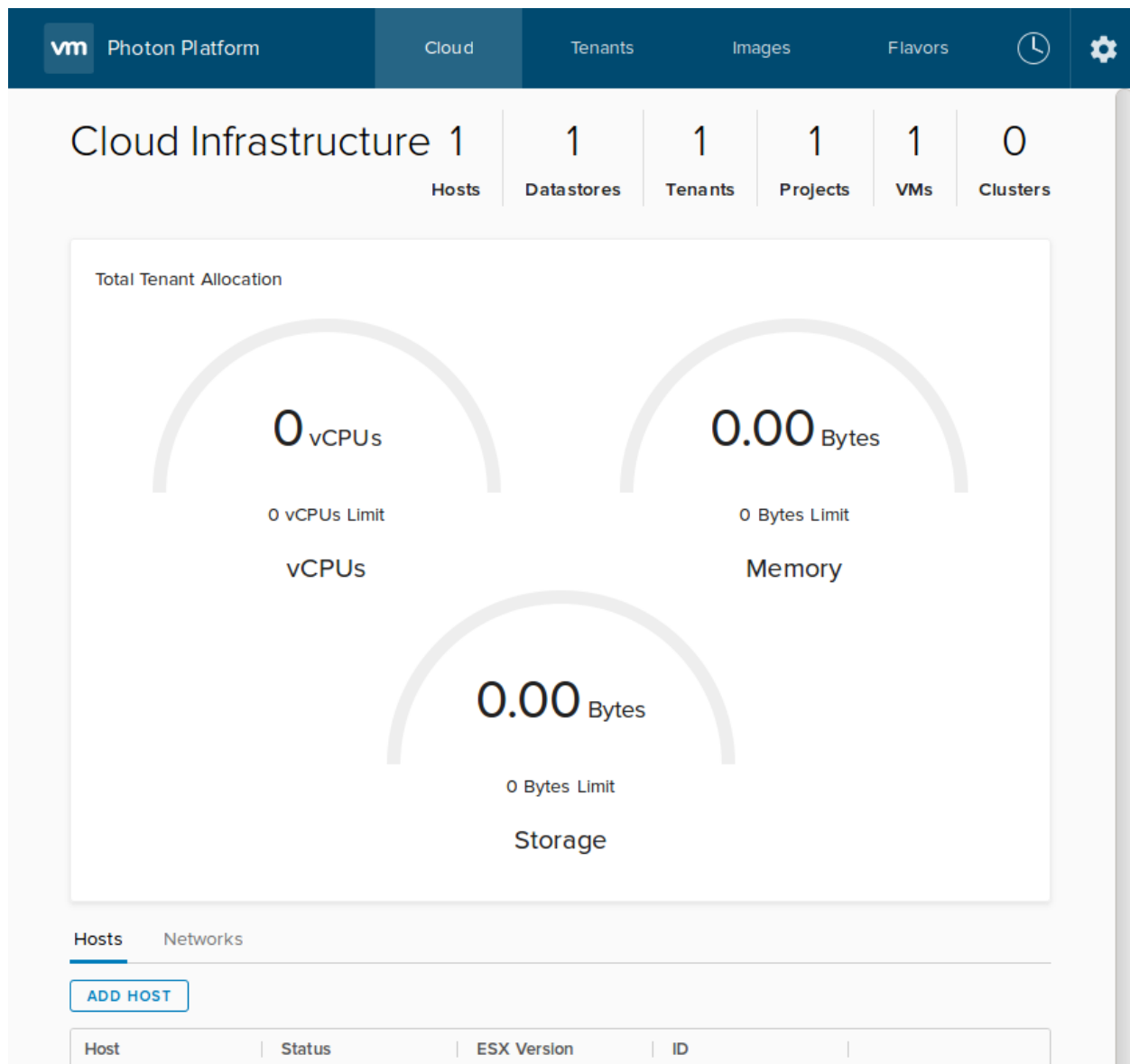


Figure 3: Photon Controller Web UI

## 4.4 An Overview of Photon Commands

This cheat sheet lists common Photon Controller commands. The commands assume that you have installed Photon Controller with authentication turned on.

**Viewing help for commands:** For information about photon commands, subcommands, and options, see the help for the Photon CLi. Examples:

```
photon --help
photon resource-ticket create --help
photon cluster create -h
```

Here's the output of `photon -h`:

NAME:

photon - Command line interface for Photon Controller

USAGE:

photon [global options] command [command options] [arguments...]

VERSION:

Git commit hash: 63a4889

COMMANDS:

auth	options for auth
system	options for system operations
target	options for target
tenant	options for tenant
host	options for host
deployment	options for deployment
resource-ticket	options for resource-ticket
image	options for image
task	options for task
flavor	options for flavor
project	options for project
disk	options for disk
vm	options for vm
network	options for network
cluster	Options for clusters
availability-zone	options for availability-zone
help, h	Shows a list of commands or help for one command

GLOBAL OPTIONS:

--non-interactive, -n	trigger for non-interactive mode (scripting)
--log-file, -l	writes logging information into a logfile at the specified path
--output, -o	Select output format
--help, -h	show help
--version, -v	print the version

**Non-interactive mode:** The `-n` option suppresses prompting but assumes that the command contains all the information required to execute successfully.

**Replacing variables in commands:** Many of the commands in this cheat sheet contain `<variables>` that you must replace. Commands containing variables are often immediately followed by examples with the variables replaced.

### 4.4.1 Setting Targets and Logging In

Find the load balancer's IP address:

```
photon deployment show
```

Connect to the IP address of the load balancer:

```
photon target set https://<ip_address>:443
```

Log in:

```
photon target login --username <username>@<oauth_tenant> --password <password>
photon target login --username pc-admin@esxcloud --password 'Your$ecret1!'
```

Check the system's status:

```
photon system status
```

#### 4.4.2 Images

Upload an image by using the `photon image create` command:

```
photon image create <local-path-to-image> -n <name> -i <replicationType>
photon image create /tmp/ubuntu14-04.ova -n ubuntu1404 -i ON_DEMAND
```

Create an image, list all your images, show information about an image, and then delete it:

```
photon image create <image_filename> -n <image_name> -i <image_type>
photon image create photon-kubernetes-vm-disk1.vmdk -n photon-kubernetes-vm.vmdk -i EAGER
photon image list
photon image show <image-ID>
photon image show 96cd7af4-f1d0-45ea-8ed1-e18bef6c05ca
photon image delete <image-ID>
photon image delete 96cd7af4-f1d0-45ea-8ed1-e18bef6c05ca
```

#### 4.4.3 Tenants and Projects

Here are some of the commands for managing tenants and resource tickets:

```
photon tenant create <name_of_tenant>
photon tenant list
photon tenant show <tenant-ID>
photon resource-ticket list
photon resource-ticket show <resource ticket name>
photon resource-ticket create
```

Here are some examples:

```
photon tenant create plato
photon resource-ticket create --tenant "plato" --name "plato-resources"
--limits "vm.memory 100 GB, vm.cpu 100 COUNT, vm 100 COUNT,
         persistent-disk 100 COUNT, persistent-disk.capacity 200 GB"
photon project create --tenant "plato" --name "plato-prjt" --resource-ticket "plato-resources"
--limits "vm.memory 100 GB, vm.cpu 100 COUNT, vm 100 COUNT,
         persistent-disk 100 COUNT, persistent-disk.capacity 200 GB"
photon tenant set "plato"
photon project set "plato-prjt"
```

#### 4.4.4 Flavors and Disks

Here are examples of how to provision resources for virtual machines or clusters with `photon flavor create`:

```

photon -n flavor create --name "vm-basic" --kind "vm"
                        --cost "vm 1 COUNT, vm.cpu 2 COUNT, vm.memory 2 GB"
photon -n flavor create --name "disk-eph" --kind "ephemeral-disk"
                        --cost "ephemeral-disk 1 COUNT"
photon -n flavor create --name "disk-persist" --kind "persistent-disk"
                        --cost "persistent-disk 1 COUNT"
photon -n flavor create --name "my-vm" --kind "vm"
                        --cost "vm 1 COUNT, vm.cpu 1 COUNT, vm.memory 2 GB"

```

#### 4.4.5 Virtual Machines and Disks

Stop a VM by its ID and create a new image from it:

```

photon vm stop bac117cb-fc32-46e0-abcd-999199c6b6d5
photon vm create-image bac117cb-fc32-46e0-abcd-999199c6b6d5 -n image-1 -r ON_DEMAND

```

More examples of creating a VM:

```

photon vm create -n vm-1 -f core-100 -d "disk-1 core-100 boot=true"
-i 5889ea6b-20ca-4706-99e8-87096d2c274
photon -n vm create --name vm-1 --flavor tiny --disks "disk-1 core-100 boot=true"
-w "ID of Network" -i "ID of Image" --affinities disk:"ID of Persistent disk"

```

Get a list of all the VMs in your project and show the information about one of them:

```

photon vm list
photon vm show <ID>
photon vm show bac117cb-fc32-46e0-abcd-999199c6b6d5

```

Here are examples of how to create a persistent disk:

```

photon disk create -n disk-2 -f core-100 -g 10
photon disk create --name persistent-disk-1 --flavor core-100
--capacityGB 10 --affinities vm:"ID of VM"

```

Attach or detach a persistent disk to or from a powered-off VM:

```

photon vm stop <VM-ID>
photon vm attach_disk <VM-ID <disk-ID>
photon vm attach_disk bac117cb-fc32-46e0-abcd-999199c6b6d5
-d dab22828-8cfe-441d-b837-b197adbc651e
photon vm detach_disk <VM-ID <disk-ID>
photon vm detach_disk bac117cb-fc32-46e0-abcd-999199c6b6d5
-d dab22828-8cfe-441d-b837-b197adbc651e

```

Delete a disk:

```

photon disk delete <disk-ID>
photon disk delete dab22828-8cfe-441d-b837-b197adbc651e

```

Here's how to upload and attach an ISO to a powered-off VM:

```

photon vm attach_iso <VM-ID> -p path -n name
photon vm attach_iso bac117cb-fc32-46e0-abcd-999199c6b6d5 -p /tmp/db.iso -n test-db

```

Operate a VM by citing its ID:

```

photon vm start bac117cb-fc32-46e0-abcd-999199c6b6d5
photon vm stop bac117cb-fc32-46e0-abcd-999199c6b6d5
photon vm suspend bac117cb-fc32-46e0-abcd-999199c6b6d5
photon vm resume bac117cb-fc32-46e0-abcd-999199c6b6d5

```

#### 4.4.6 Clusters

Here are examples of how to establish resources for a Kubernetes cluster:

```
photon image create photon-kubernetes-vm-disk1.vmdk -n photon-kubernetes-vm.vmdk -i EAGER
photon image list
photon deployment list
photon deployment enable-cluster-type <deployment_ID> -k KUBERNETES -i <Kubernetes_image_ID>
```

Here's how to create a Kubernetes cluster. Replace the example IP addresses with those from your Photon Controller environment. The IP address for the `master-ip` option should contain the static IP address that you want to assign to the Kubernetes cluster. The `etcd` option should also contain a static IP address.

```
photon cluster create -n kube-socrates -k KUBERNETES --master-ip 198.51.100.85
--etcd1 198.51.100.86 --container-network 192.0.2.0/16 --dns 198.51.100.1
--gateway 198.51.100.253 --netmask 255.255.0.0
```

More `photon cluster` commands:

```
photon cluster list
```

To get the Kubernetes master node IP address, use `show`:

```
photon cluster show <cluster-id>
```

See all the VMs in a cluster:

```
photon cluster list_vms <cluster-ID>
```

Delete it:

```
photon cluster delete <cluster-ID>
```

#### 4.4.7 Availability Zones

Create an availability zone, set it as the default, and create a VM in it:

```
Photon availability_zone create --name "zone-name"
photon host set_availability_zone <hostID> <availabilityZoneID>
photon -n vm create --name vm-2 --flavor core-200 --disks "new-disk core-200 boot=true"
--affinities "availabilityZone:UUIDofAZ"
```

#### 4.4.8 Authentication

View your token:

```
photon auth show-login-token
```

Example output:

```
Login Access Token:
Subject: faulkner@esxcloud
Groups: esxcloudgroup11, esxcloudEveryone
Issued: 2016-11-30 04:02:15.00
Expires: 2016-11-30 04:07:15.00
Token: ...
```

## 4.5 Working with ESXi Hosts

Here's how you can use the `photon host` command or the Photon web interface to work with ESXi hosts. You can, for example, add ESXi hosts to your Photon Controller cluster by using the `photon host create` command. Adding ESXi hosts to the cluster can expand the cluster's capacity for computation and storage so that you can run more VMs or create larger Kubernetes clusters.

### 4.5.1 Adding an ESXi Host to a Photon Controller Cluster

You can add an ESXi host through the Photon Platform web interface.

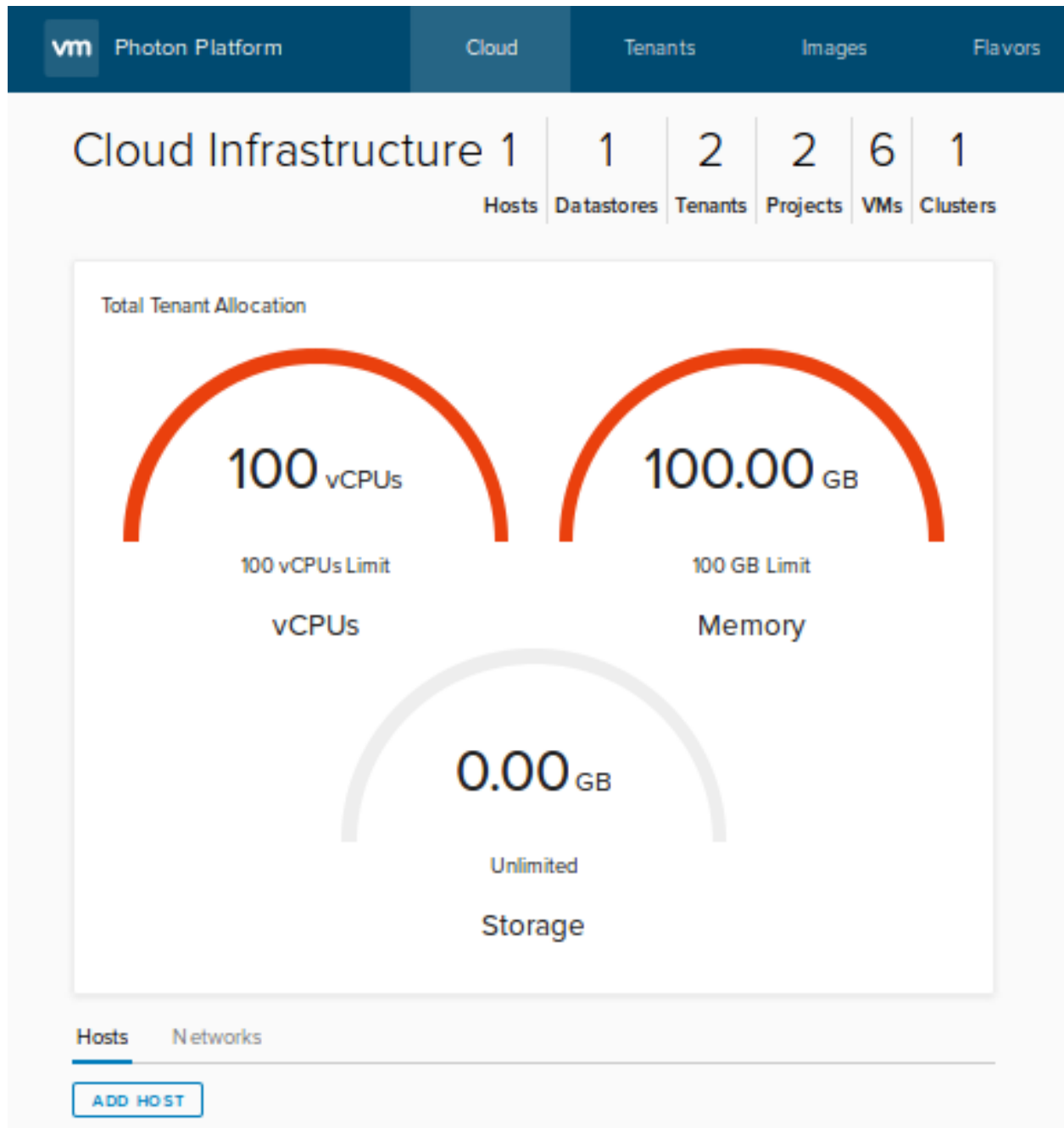


Figure 4: The Cloud Page in the web interface



You can also add an ESXi host by using the `photon host create` command:

```
photon host create --username root --password ESXiHostPw --address <ip-address-of-new-ESXi-host>
```

Here's the usage information the `photon host create` command:

NAME:

```
photon host create - Create a new host
```

USAGE:

```
photon host create [command options] [arguments...]
```

OPTIONS:

```
--username, -u      username to create host
--password, -p      password to create host
--address, -i       ip address of the host
--availability_zone, -z availability zone of the host
--tag, -t           tag for the host
--metadata, -m      metadata for the host
--deployment_id, -d deployment id to create host
```

#### 4.5.2 Viewing Information about ESXi Hosts

List the ESXi hosts in a Photon Controller cluster and show their status:

```
photon host list
ID                State IP                Tags
137d4e5a5437f85fea900  READY  198.51.100.44  MGMT CLOUD
Total: 1
```

Show complete information for a host by citing its ID:

```
photon host show 137d4e5a5437f85fea900
Host ID: 137d4e5a5437f85fea900
Username:      root
IP:            198.51.100.44
Tags:          [MGMT CLOUD]
State:         READY
Metadata:      map[MANAGEMENT_VM_DISK_GB_OVERWRITE:80
MANAGEMENT_NETWORK_DNS_SERVER:198.51.98.1
MANAGEMENT_NETWORK_NETMASK:255.255.0.0
MANAGEMENT_NETWORK_IP:198.51.100.207
MANAGEMENT_NETWORK_GATEWAY: 198.51.100.253
MANAGEMENT_DATASTORE:datastore1
MANAGEMENT_VM_CPU_COUNT_OVERWRITE:4
MANAGEMENT_PORTGROUP:VM Network
MANAGEMENT_VM_MEMORY_MB_OVERWRITE:6144]
AvailabilityZone:
Version:       6.0.0
```

List the VMs on an ESXi host and show their state by citing the ID of the host:

```
photon host list-vms 137d4e5a5437f85fea900
ID                Name                State
1305ecbd-8717-4dad-bc4f-1e7b6cebb897  Photon-OS          STARTED
19d197fc-2653-466f-bb7f-5f03a5a6bdb1  ec-mgmt-10-118-100-448095e  STARTED
1eba91ba-c926-41dc-adb1-4b82e0797c4b  master-7a0c629b-52dc-49c9-b58f-f37100948487  STARTED
```

```

71b20ca1-5055-4530-8218-e0ca0730e79d worker-7daa6a2c-39f9-4834-be33-4ccac548f8da STARTED
85c9325b-ed0f-4f63-8bd6-8c581d4593fa etcd-96cae740-3c65-4ac8-96e9-c6b170990934 STARTED
ebce1024-1282-49d5-a4dc-930d40923d18 Photon-OS-2 STARTED
Total: 6
STARTED: 6

```

Show the tasks on a host:

```

photon host tasks 137d4e5a5437f85fea900
Task                Start Time          Duration
137d4e5a5437f85fee398 2016-12-12 04:44:31.00 00:00:10
CREATE_HOST, COMPLETED
You can run 'photon task show <id>' for more information
Total: 1

```

There are additional commands for managing the ESXi hosts in a Photon Controller cluster:

```

photon host --help
NAME:
    photon host - options for host
USAGE:
    photon host command [command options] [arguments...]
COMMANDS:
    create          Create a new host
    delete          Delete a host with specified id
    list            List all the hosts
    show            Show host info with specified id
    list-vms        List all the vms on the host
    set-availability-zone Set host's availability zone
    tasks           Show tenant tasks
    suspend         Suspend host with specified id
    resume          Resume host with specified id
    enter-maintenance Host with specified id enter maintenance mode
    exit-maintenance Host with specified id exit maintenance mode
    help, h         Shows a list of commands or help for one command
OPTIONS:
    --help, -h  show help

```

## 5 Working with Tenants, Projects, and VMs

### 5.1 About Tenants, Resource Tickets, and Projects

The basic elements of Photon Controller's multitenancy model are tenants, resource tickets, and projects.

#### 5.1.1 Tenants

The top-level tenancy element in Photon Controller is called a tenant. A tenant has resource tickets and projects.

A tenant administrator can manage the projects in a tenant. A tenant has a set of Lightwave security groups that specifies the set of users who can act as tenant administrators.

### 5.1.2 Resource Tickets

Resource tickets represent allocations of physical or virtual objects. You allocate tickets under a tenant by using the same free-form syntax as flavor definitions. You can think of a resource ticket as a quota.

Before you create a project, the tenant must have at least one resource ticket.

For example, a resource ticket might grant a quota of up to “100 virtual CPUs and 1000 GB of RAM.” The quota does not guarantee that the resources are available, which depends on your hardware. Rather, the quota limits the resources available to the tenant.

When creating a resource ticket, you must set at least one limit, such as virtual CPUs.

There is a strong relationship between flavors and resource tickets. Both let you specify nearly arbitrary values that are accounted for. (It is almost arbitrary because VM flavors must specify the `vm.cpu` and `vm.memory` costs.) For example, a resource ticket may specify the following limits:

- `vm.cpu COUNT 100`: The user may have up to 100 virtual CPUs
- `vm.calories COUNT 3000`: The user may use up to 3000 calories. This example uses calories as a user-defined accounting metric.

A VM flavor may specify the following:

- `vm.cpu COUNT 2`: (This VM will use 2 virtual CPUs)
- `vm.memory 2 GB`: (This VM will use 2 GB of RAM)
- `vm.calories 10 COUNT`: (This VM will use 10 calories)

Using this flavor with this resource ticket, a user could create 50 VMs (since each uses two CPUs). Because the resource ticket does not specify memory, the user’s memory usage remains unrestricted.

### 5.1.3 Projects

Projects represent individual users or groups under a tenant. A project is assigned a subset of the resources from a resource ticket at creation time. For instance, if a resource ticket allows 100 virtual CPUs, a project could allow anywhere between 0 and 100 CPUs from the resource ticket. Resource tickets, in other words, can be subdivided among projects.

## 5.2 Working with Tenants

In Photon Controller’s multitenancy model, tenants segregate users. A tenant can be a developer, a team of engineers, a business unit, or any other user or group. Tenants let you isolate users, control access to the API, regulate resources, and dynamically allocate resources to transient workloads.

As a system administrator, you can assign resource tickets, or quotas, to the tenants you create. A tenant administrator can then create projects and allocate resources from the quota to the projects and their users.

Only a system administrator can create a tenant administrator by assigning a user to a security group for tenants. The tenant administrator can manage projects, resource tickets, and other objects owned by the tenant. Either a tenant administrator or a system administrator can create project users, who can manage project resources, such as VMs, disks, and images.

### 5.2.1 Creating a Tenant

As a system administrator, you can create a tenant by running the following command in the Photon command-line utility on your workstation:

```
photon tenant create <name_of_tenant>
```

## 5.2.2 Commands for Working with Tenants

List tenants:

```
photon tenant list
```

Show information about a tenant:

```
photon tenant show <tenant-ID>
```

List resource tickets:

```
photon resource-ticket list
```

Show the limits and project allocations of a ticket:

```
photon resource-ticket show "resource ticket name"
```

List projects:

```
photon project list
```

Show information about a project, such as its limits and consumption:

```
photon project show <project-ID>
```

## 5.2.3 Related API Endpoints

/tenants

/projects

/resource-tickets

## 5.3 Creating a Project

A project carves out a set of resources for a tenant. A project requires a resource ticket, and the project can consume all or part of the resources defined by its resource ticket. For each resource limit defined by the ticket, you specify a limit for that resource when creating the project.

To create a project, you must be logged in as a tenant administrator or a project user.

### 5.3.1 Project Commands

You can use the `photon project` command to create and manage projects:

```
photon project -h
```

NAME:

```
photon project - options for project
```

USAGE:

```
photon project command [command options] [arguments...]
```

COMMANDS:

create	Create a new project
delete	Delete project with specified id
show	Show project info with specified id
get	Get project in config file
set	Set project in config file
list	List all projects
tasks	List all tasks related to the project
set_security_groups	Set security groups for a project

help, h            Shows a list of commands or help for one command

The photon project create command includes the following options:

```
photon project create -h
```

NAME:

```
photon project create - Create a new project
```

USAGE:

```
photon project create [command options] [arguments...]
```

OPTIONS:

```
--name, -n            Project name
--limits, -l          Project limits(key,value,unit)
--percent, -p "0"      Project limits(percentage of resource-ticket)
--tenant, -t          Tenant name for project
--resource-ticket, -r   Resource-ticket name for project
--security-groups, -g   Security Groups for project
```

### 5.3.2 Example of How To Create a Tenant and a Project

A project is tied to a tenant. A tenant is a unit of administrative control allocated a pool of resources for projects, such as a Kubernetes cluster. The following sequence of commands creates a tenant, gives it a pool of resources, and creates a project with a ticket to use all the resources in the pool.

```
photon tenant create plato
```

```
photon resource-ticket create --tenant "plato" --name "plato-resources"
```

```
--limits "vm.memory 100 GB, vm.cpu 100 COUNT, vm 100 COUNT,
persistent-disk 100 COUNT, persistent-disk.capacity 200 GB"
```

```
photon project create --tenant "plato" --name "plato-prjt"
```

```
--resource-ticket "plato-resources"
--limits "vm.memory 100 GB, vm.cpu 100 COUNT, vm 100 COUNT,
persistent-disk 100 COUNT, persistent-disk.capacity 200 GB"
```

Many photon subcommands take place in the context of a tenant or a project. To simplify commands for working with a tenant or a project, you can set each to persist as your default. Here's an example:

```
photon tenant set "plato"
```

```
photon project set "plato-prjt"
```

Photon saves the values that you supply for the `set` command in the file named `.photon` in your home directory. To change your default project or tenant, run the `set` command again with a new tenant or project name.

## 5.4 Uploading Images

You can upload images to Photon Controller to create virtual machines and clusters. An image can be an OVA, vmdk, or ISO file. After you upload an OVA for Ubuntu 16.04, for example, you can create VMs from it on demand.

Photon Controller lets you seed the system with images that can be shared among tenants. As a tenant, you can also create your own images without sharing them with other tenants.

Before you can generate a VM or disk from an image, however, you must create a [flavor](#) for it. A flavor specifies a template for the resources and costs that a disk or VM image consumes. See [Flavors](#).

For instructions on how to upload and enable the Kubernetes image, see [Creating a Kubernetes Cluster](#).

### 5.4.1 Setting the Target

To work with Photon Controller from a workstation, you first set the target by using the Photon CLI. Setting the target establishes a connection to Photon Controller. The following command assumes that you set up Photon Controller to [authenticate](#) users.

```
photon target set https://<ip_address>:443
```

If you enabled the load balancer during deployment, the IP address in the command should be that of the load balancer, not a management node.

### 5.4.2 Uploading Images

To upload images, run the command `photon image create` command, replacing `<type>` with either `ON_DEMAND` or `EAGER`:

```
photon image create <local_file_path_to_image> -n <name> -i <type>
```

Here's an example:

```
photon image create /tmp/ubuntu16-04.ova -n ubuntu1604 -i ON_DEMAND
```

### 5.4.3 Eager Images and On-Demand Images

Photon Controller replicates an eager image to each cloud host's datastore when it is uploaded, making it immediately available across the system. In contrast, Photon Controller replicates an on-demand image to other cloud hosts only when a tenant creates a VM from it. An eager image produces VMs faster but consumes more storage space; an on-demand image produces VMs slower but takes less storage space.

### 5.4.4 Viewing Images

The `photon image list` command returns a list of images and their IDs.

The `photon image show <image_ID>` command displays details about an image, including its size, settings, and replication type.

### 5.4.5 Deleting Images

You can delete an image like this:

```
photon image delete <image_ID>
```

## 5.5 Creating Images

An image represents a category of virtual machine or disk. You can create an image by uploading an OVA or VMDK file to the system's shared image store. You can also create an image by capturing an instance of a powered-off VM or unattached disk. The system administrator typically provisions the system with a variety of images to share across all tenants.

Tenants can also create images that are not shared with other tenants. An image does not include any information about the resources the virtual machine or disk consumes. Before you can use a VM or disk image, a system administrator must create one or more flavors of the right kind.

For instructions on how to upload and enable the Kubernetes image, see [Creating a Kubernetes Cluster](#).

### 5.5.1 Creating Images

Here's how to create an image:

```
photon image create <image_filename> -n <image_name> -i <image_type>
```

### 5.5.2 Viewing the List of Images

You can verify the image was created properly by examining the output of the following command:

```
photon image list
```

### 5.5.3 Uploading an OVA File to Create an Image

One way to create a new image is by uploading an OVA file. In the following command, replace `replicationType` with `ON_DEMAND` or `EAGER`.

```
photon image create local-path-to-OVA -n name -i replicationType
```

Here's an example:

```
photon image create /tmp/ubuntu14-04.ova -n ubuntu1404 -i ON_DEMAND
```

When Photon Controller uploads an image, it returns an ID for it. You can append it to the `photon image show` command to display information about the image; example:

```
photon image show 96cd7af4-f1d0-45ea-8ed1-e18bef6c05ca
```

### 5.5.4 Creating an Image from a VM

Here's an example of how you can create an image from a powered-off VM. First, stop the VM from which you want to create an image:

```
photon vm stop 96cd7af4-f1d0-45ea-8ed1-e18bef6c05ca
```

And then create the image:

```
photon vm create-image 96cd7af4-f1d0-45ea-8ed1-e18bef6c05ca  
-n image-1 -r ON_DEMAND
```

### 5.5.5 Deleting an Image

You can delete an image if you belong to a security group that gives you permission to do so.

Here's the command:

```
photon image delete ID
```

Example:

```
photon image delete 96cd7af4-f1d0-45ea-8ed1-e18bef6c05ca
```

## 5.6 Replicating Images in Datastores

### 5.6.1 Image Datastores

An image datastore is an ESXi datastore that is shared between multiple ESXi hosts. (Technically it does not have to be shared, but it usually is.)

### 5.6.2 Image Replication and Datastores

A typical installation is to have a small set of image datastores shared between hosts (typically one image datastore per host) and one or more local datastores on each host.

When you create an image in Photon Controller, you specify that it is `ON_DEMAND` or `EAGER`. In either case, it will be transferred to all image datastores in the system. If you specify `EAGER`, it will also be transferred to all the other datastores in the system. This can result in faster VM creation, but it uses more storage. If you specify `ON_DEMAND` and the ESXi host that was chosen to host the VM has a datastore in addition to the image datastore, the image will be copied from the image datastore to that datastore. (If there is more than one local datastore, a single datastore is chosen.) `ON_DEMAND` uses less storage, but it takes longer to create the VM.

When you configure Photon Controller, you can specify the set of allowed datastores. If you do not specify the set, Photon Controller will use all datastores that it find. Images will only be replicated or copied to the set of allowed datastores. Please note that datastores attached at runtime (not just when the system is created) will be found and will be used if they are allowed.

It is acceptable to have exactly one datastore in your deployment, if it is shared across all hosts. This is a simple setup because it does not require any replication, but it does require that your datastore can support the the load that will be placed on it.

## 5.7 Creating Flavors

A flavor is a named collection of *costs* for a VM or disk image. Flavors are closely related to [resource tickets](#): The costs that a flavor specifies are subtracted from the total allocation in the resource ticket you are using.

To create VM or disk, you must specify both an image and a flavor.

### 5.7.1 Types of flavors

There are three types of flavors:

- *vm*: A VM flavor specifies the costs in creating a VM.
- *ephemeral-disk*: Ephemeral disks are used for your boot disks when creating a VM.
- *persistent-disk*: Persistent disks can be attached to your VMs and can persist after a VM is removed.

### 5.7.2 Flavor Costs

You can make any set of costs for a flavor that you want as long as you follow some rules:

- A single cost is a number (fractional numbers are allowed) and a unit. Units can be B, KB, MB, GB (for size) or COUNT (for totals).
- VM flavors must specify both the `vm.cpu` cost (this is a COUNT), and the `vm.memory` cost (units are B, KB, MB, or GB). For example, a VM flavor that specifies 2 CPUs and 4 GB of memory looks specifies costs as “`vm.cpu 2 COUNT, vm.memory 4 GB`”. With the Photon CLI, it looks like this:



```
photon -n flavor create --name "vm-cpu-memory" --kind "vm"
                        --cost "vm.cpu 2 COUNT, vm.memory 4 GB"
```

- If you want to limit the total number of VMs a user can create, you'd make a "vm COUNT" cost. Extending the example above, the costs would be: "vm 1 COUNT, vm.cpu 2 COUNT, vm.memory 4 GB". This doesn't have to be named "vm", but it's a conventional way to do it. There must be a corresponding limit in the resource ticket.
- If you want to specify other costs, you can do so. For example, you can make a "vm.calories COUNT" cost. The meaning of this is entirely up to you. Photon Controller will ensure the user doesn't exceed their calorie limit, but will not assign special meaning when managing the VM.
- When a VM is created, it's boot disk must specify a flavor, and that flavor must specify a cost. A typical disk flavor only specifies the count. For ephemeral disks, the cost would be "ephemeral-disk 1 COUNT", for persistent disks the cost would be "persistent-disk 1 COUNT"
- Disk flavors must not specify the capacity: this is calculated from the size of the boot disk image or the size of the disk that is created.

### 5.7.3 Creating Flavors

You create flavors by using the `photon flavor create` command. You'll be prompted for a few details.

```
photon flavor create
Flavor name: tiny
Flavor kind (persistent-disk, ephemeral-disk, or vm): vm
```

```
Limit 1 (ENTER to finish)
Key: vm.cpu
Value: 1
Unit: COUNT
```

```
Limit 2 (ENTER to finish)
Key: vm.memory
Value: 512
Unit: MB
```

```
Limit 3 (ENTER to finish)
Key:
Creating flavor: 'tiny', Kind: 'vm'
```

```
Please make sure the limits below are correct:
1: vm.cpu, 1, COUNT
2: vm.memory, 512, MB
Are you sure [y/n]? y
Using target 'https://10.18.155.101:443'
Created flavor ID: d168da40-71d2-4ccf-ac94-eda8fa3b081c
```

### 5.7.4 List All Flavors

You can list all flavors configured on the system; abridged example:

```
photon flavor list
Using target 'https://10.18.155.101:443'
```

ID	Name	Kind	Cost
d168da40-71d2-4ccf-ac94-eda8fa3b081c	tiny	vm	vm.cpu 1 COUNT

vm.memory 512 MB

Total: 6

### 5.7.5 Show Flavor Details

To see details about the flavor, run the `photon flavor show <id>` command:

```
photon flavor show d168da40-71d2-4ccf-ac94-eda8fa3b081c
Using target 'https://10.18.155.101:443'
Flavor ID: d168da40-71d2-4ccf-ac94-eda8fa3b081c
  Name: tiny
  Kind: vm
  Cost: [vm.cpu 1 COUNT vm.memory 512 MB]
  State: READY
```

### 5.7.6 Create Cluster with New Flavor

If you want to shrink the instances that make up your Mesos cluster. After going through the process of [creating](#) your new flavor, re-create the cluster using the new sizing.

```
photon cluster create -n Mesos2 -k MESOS <...> --vm_flavor tiny
```

## 5.8 Provisioning Virtual Machines

### 5.8.1 Creating a Virtual Machine from an Image

You create a virtual machine by specifying an image, a name, a flavor, and an ephemeral boot disk. Unlike persistent disks, which must be created before they can be attached to a VM, ephemeral disks are created when the VM is created and discarded when the VM is removed.

When you create a VM, you combine an image and a flavor (CPU and memory resource limits) with an ephemeral boot disk. You can attach other disks to the VM during or after creation.

To create a VM in interactive mode, use the `photon vm create` command with no options. The command prompts you for the VM name, flavor, and source image ID. Example:

```
photon vm create
VM name: vm-1 VM
flavor: core-100
Image id: 5889ea6b-20ca-4706-99e8-f87096d2c274
Disk 1 (ENTER to finish):
Name: disk-1
Flavor: core-100
Boot disk? [y/n]: y

Disk 2 (ENTER to finish):
Name:
Creating VM: vm-1 (core-100)
Source image id: 5889ea6b-20ca-4706-99e8-f87096d2c274
Disks: 1: disk-1, core-100, boot
Are you sure? y
VM 'bac117cb-fc32-46e0-abcd-999199c6b6d5' created
```

You can also supply this information when you run the `photon vm create` command with the non-interactive option, `-n`. Example:

```
photon vm create -n vm-1 -f core-100 -d "disk-1 core-100 boot=true"
-i 5889ea6b-20ca-4706-99e8-87096d2c274
{:name=>"vm-1"}
Creating VM: vm-1 (core-100)
Source image id: 5889ea6b-20ca-4706-99e8-f87096d2c274
Disks: 1: disk-1, core-100, boot
```

### 5.8.2 Getting Information about VMs

Get a list of all the VMs in your project:

```
photon vm list
```

Get information about a VM:

```
photon vm show <ID>
photon vm show bac117cb-fc32-46e0-abcd-999199c6b6d5
```

You can use the `photon vm networks` command to get more information about a VM's network connections:

```
photon vm networks 3db6bd84-4fe2-413f-b502-f60f75f74f2d
```

### 5.8.3 Attaching Persistent Disks to VMs

Persistent disks endow virtual machines with resources for a project. You can attach a persistent disk to a VM, but after you discard the VM, the disk persists. Disks stick with a project, and they cannot be used by other projects.

Photon Controller categorizes persistent disks as a type of [flavor](#), subtracting each persistent disk from the resources allocated to a project.

The following example commands assume you have set the default project.

Here are the commands to create persistent disks. The commands assume you have set the default project.

```
photon disk create
```

The command prompts you for the disk name, flavor, and capacity. Example:

```
DISK name: disk-2
DISK flavor: core-100
DISK capacity in GB: 10
```

The following command creates the same persistent disk by specifying the disk's properties in the command:

```
photon disk create -n disk-2 -f core-100 -g 10
```

Here's how to delete a disk:

```
photon disk delete <disk-ID>
```

Example:

```
photon disk delete dab22828-8cfe-441d-b837-b197adbc651e
```

A persistent disk can be attached to a powered-off VM:

```
photon vm stop <VM-ID>
photon vm attach_disk <VM-ID <disk-ID>
photon vm detach_disk <VM-ID <disk-ID>
```

Examples:

```
photon vm attach_disk bac117cb-fc32-46e0-abcd-999199c6b6d5
                        -d dab22828-8cfe-441d-b837-b197adbc651e
```

```
photon vm detach_disk bac117cb-fc32-46e0-abcd-999199c6b6d5
                        -d dab22828-8cfe-441d-b837-b197adbc651e
```

### 5.8.4 Attaching an ISO to a VM

Here's how to upload and attach an ISO to a powered-off VM:

```
photon vm attach_iso <VM-ID> -p path -n name
```

Example:

```
photon vm attach_iso bac117cb-fc32-46e0-abcd-999199c6b6d5 -p /tmp/db.iso -n test-db
```

### 5.8.5 Operating a Virtual Machine

You can start, stop, restart, suspend, and resume a virtual machine by citing its ID. Examples:

```
photon vm start bac117cb-fc32-46e0-abcd-999199c6b6d5
photon vm stop bac117cb-fc32-46e0-abcd-999199c6b6d5
photon vm suspend bac117cb-fc32-46e0-abcd-999199c6b6d5
photon vm resume bac117cb-fc32-46e0-abcd-999199c6b6d5
```

## 5.9 Creating a Network

Here's how you can use the `photon network` command to work with networks on the ESXi hosts in a Photon Controller cluster.

With the Photon command-line utility, you can view the options of the `photon network` command like this:

```
photon network --help
NAME:
    photon network - options for network
USAGE:
    photon network command [command options] [arguments...]
COMMANDS:
    create    Create a new network
    delete   Delete a network
    list      List networks
    show      Show specified network
    set-default Set default network
    help, h   Shows a list of commands or help for one command
OPTIONS:
    --help, -h  show help
```

### 5.9.1 Checking the Default Network

With a typical installation of Photon Controller, there is one default network, usually `VM Network`. You can check for a default network by running the following command:

```
photon network list
```

ID	Name	State	PortGroups	Descriptions	IsDefault
----	------	-------	------------	--------------	-----------

```
f858e2fd5437c42096940  vm-network  READY  [VM Network]  photon network list  true
Total: 1
```

And then you can view more information about it by citing its ID:

```
photon network show f858e2fd5437c42096940
Network ID: f858e2fd5437c42096940
  Name:          vm-network
  State:         READY
  Description:   photon network list
  Port Groups:  [VM Network]
  Is Default:   true
```

### 5.9.2 Creating a New Network

Adding a new network and associating it with a Photon Controller project can set aside networking resources for the project's virtual machines and their users.

To create a network in the Photon Platform web interface, on the **Cloud** page, click **Networks** and then click **New Network**:

You can also add a new network by running the `photon network create` command. It interactively prompts you to enter the name, description, and port groups of the network that you want to create. Photon Controller assigns an ID to the network.

Here is the usage information for the `photon network create` command:

```
photon network create --help
NAME:
  photon network create - Create a new network
USAGE:
  photon network create [command options] [arguments...]
OPTIONS:
  --name, -n          Network name
  --description, -d   Description of network
  --portgroups, -p    PortGroups associated with network (only for physical network)
  --routingType, -r   Routing type for network (only for software-defined network).
                     Supported values are: 'ROUTED' and 'ISOLATED'
  --size, -s          Size of the private IP addresses (only for software-defined network)
  --staticIpSize, -f  Size of the reserved static IP addresses
                     (only for software-defined network)
  --projectId, -i    ID of the project that network belongs to
                     (only for software-defined network)
```

### 5.9.3 Setting the Default Network

Once a network is created, you can set it as your default by citing its ID; example:

```
photon network set-default f858e2fd5437c42096940
Are you sure [y/n]? y
SET_DEFAULT_NETWORK completed for 'subnet' entity f858e2fd5437c42096940
```

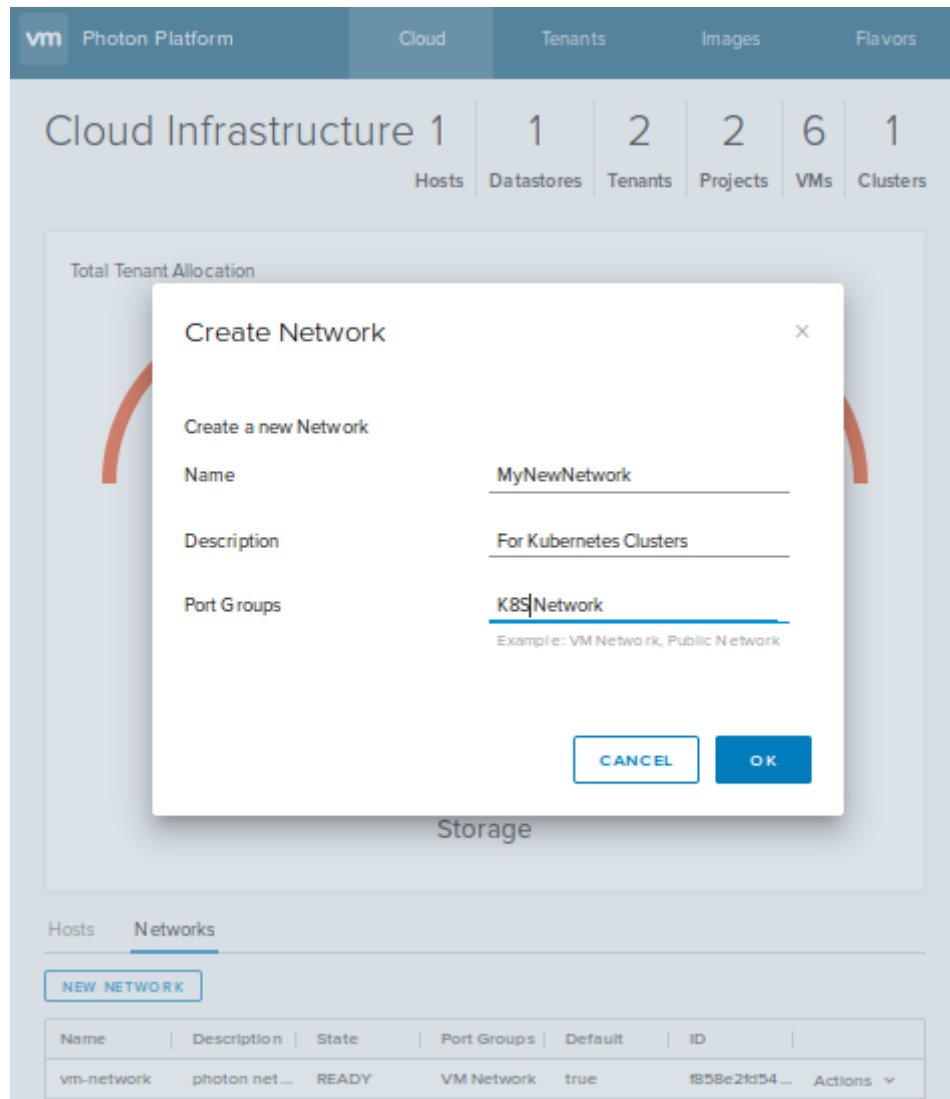


Figure 5: Adding a Network

## 5.10 Using Photon OS

Photon OS is an open-source minimalist Linux operating system from VMware. The operating system is optimized to work as a container run-time environment for VMware vSphere deployments, cloud-computing platforms, and cloud-native applications.

### 5.10.1 Producing a Photon OS VM in Photon Controller

You can download the OVA for the minimal version of Photon OS from Bintray and deploy it in Photon Controller in a matter of seconds. Here's how:

On your workstation, download the OVA for the minimal version of Photon OS from the following URL:

<https://bintray.com/vmware/photon/ova>

Now open a web browser and connect to the Photon Controller web interface:

<http://<ip-address-of-load-balancer>>

In the Photon Controller web interface, click **Images**, and then click **Upload image**.

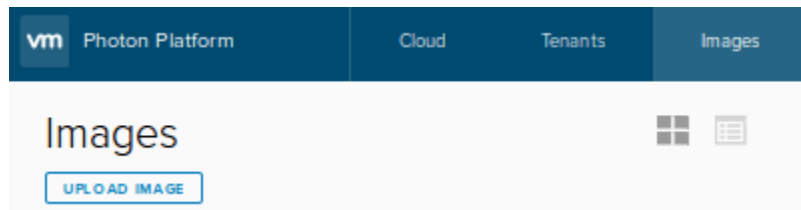


Figure 6: Upload Image

Click **Browse**, find the OVA for Photon OS that you downloaded, and then click **OK**. The file name for the Photon OS OVA looks something like this, though the version and build numbers might be different:

`photon-custom-hw11-1.0-13c08b6.ova`

To see the progress of the upload, click the image of the clock—the Activity Stream—in the Photon Controller navigation bar.

After the image uploads, you can start a VM running on Photon OS by clicking **Tenants** in the navigation bar, selecting a tenant in the list, selecting a project in the list of the tenant's projects, and then clicking **New Virtual Machine**:

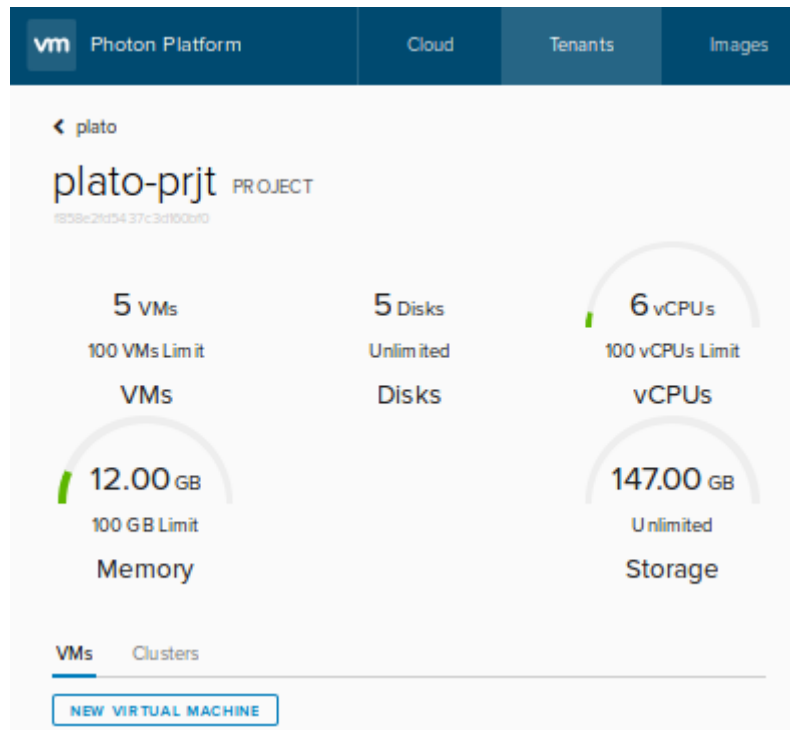


Figure 7: New Virtual Machine



Fill out the form to create a new VM. In the drop down for **Image**, select the image for Photon OS. For the flavor, you can leave the default, `cluster-other-vm`. Here's an example:

## New Virtual Machine

Create a new virtual machine in the project 'plato-prjt'

Virtual machine name	Photon-OS-2
VM network	Default ▾
VM flavor	cluster-other-vm ▾
Image	photon-custom-hw11-1.0-13c08b ▾
Boot disk name	phos2
Boot disk flavor	cluster-vm-disk ▾

Figure 8: The Form for a New VM

After Photon Controller creates the VM, start it by clicking **Actions** in the list of VMs under the project, and then click **Power On**.

You can now connect to the VM and use it in your clusters or for other purposes. On Photon OS, the default password for the root account is **changeme**, and you must change it when you first login. For security, Photon OS forbids common dictionary words for the root password.

### 5.10.2 Docker Containers on Photon OS

Photon OS includes the open source version of Docker. With Docker, Photon OS becomes a Linux run-time host for containers—that is, a Linux cloud container.

On Photon OS, the Docker daemon is enabled by default. To view the status of the daemon, run this command:

```
systemctl status docker
```

Docker is loaded and running by default on the full version of Photon OS. On the minimal version, it is loaded but not running by default, so you have to start it:

```
systemctl start docker
```

To obtain information about Docker, run this command as root:

```
docker info
```

After you make sure that docker is enabled and started, you can, for example, run the following docker command as root to create a container running Ubuntu 14.04 with an interactive terminal shell:

```
docker run -i -t ubuntu:14.04 /bin/bash
```

Photon OS also enables you to run a docker container that, in turn, runs Photon OS:

```
docker run -i -t photon /bin/bash
```

For information about working with Photon OS, see the [Photon OS Administration Guide](#). It includes sections on initializing Photon OS with cloud-init, running Docker containers, and working with Kubernetes.

## 6 Deploying Clusters

### 6.1 Creating a Kubernetes Cluster

#### Table of Contents

- [Introduction](#)
- [Requirements](#)
- [Obtaining, Uploading, and Enabling the Kubernetes Image](#)
- [Creating a Tenant and a Project](#)
- [Creating Resources for Use in the Cluster](#)
- [Creating Resources for Containerized Applications](#)
- [Setting Up a Network for Use with the Cluster](#)
- [Creating the Kubernetes Cluster](#)
- [Checking the Cluster's Resources and Status](#)
- [Opening the Kubernetes Dashboard](#)
- [Deploying an nginx Web Server](#)
- [Troubleshooting Cluster Creation](#)
- [Related](#)

### 6.1.1 Introduction

Setting up a Kubernetes cluster for a web application demonstrates the power of Kubernetes as a service.

Deploying a Kubernetes cluster involves two main steps: creating resources for a tenant and spinning them up into a cluster with the `photon cluster create` command. On Photon Controller, the following primitives are the building blocks of clusters:

- Images
- Virtual machines
- Disks
- Resource tickets

### 6.1.2 Requirements

A Kubernetes cluster carries several requirements:

- A static IP address to assign as the master IP address of the cluster.
- At least one static IP address for an etcd node in the cluster. For high availability of etcd, you'll need three static IP addresses.
- The [Kubernetes image](#) for Photon Controller.

The instructions assume that you are using a Linux workstation with the Photon Controller CLI connected to a deployment of Photon Controller on ESXi. You can also use a Mac or Windows workstation with the Photon Controller CLI installed, but you will have to adapt some of the commands to your operating system and environment.

For information about photon commands, subcommands, and options, see the help for the Photon Controller CLi on your Linux workstation; examples:

```
photon --help
photon resource-ticket create --help
photon cluster create -h
```

### 6.1.3 Obtaining, Uploading, and Enabling the Kubernetes Image

Download the Kubernetes disk image for Photon Controller from the following URL to the workstation on which you are running the Photon Controller command-line utility:

<https://github.com/vmware/photon-controller/releases>

The Kubernetes image is packaged as an OVA; it has a file name that looks like this, though the version and build numbers might be different:

`kubernetes-1.4.3-pc-1.1.0-5de1cb7.ova`

Upload the the Kubernetes image to Photon Controller by running the following commands as the system administrator, replacing the variables with the IP address of the load balancer for the Photon Controller management plane:

```
photon target set https://mgmt-ip-address:443
photon image create kubernetes-1.4.3-pc-1.1.0-5de1cb7.ova -n kube1 -i EAGER
```

The upload takes a few minutes. When it finishes, enable the image by obtaining its ID with the `photon image list` command and using it to replace the variable in the `enable-cluster-type` command:

```
photon image list
photon deployment enable-cluster-type -k KUBERNETES -i <Kubernetes_image_ID>
```

As the Photon Controller system administrator, you typically need to enable a cluster type only once. Enabling the cluster type sets Photon Controller to use the given image to deploy each of your Kubernetes clusters. You should, however, disable the cluster type for a given image before you delete the image.

#### 6.1.4 Creating a Tenant and a Project for the Cluster

A tenant is a unit of administrative control allocated a [quota](#) for projects, such as a Kubernetes cluster. The following sequence of commands creates a tenant, gives it a pool of resources, and creates a project with a ticket to use all the resources in the pool.

```
photon tenant create plato
photon resource-ticket create --tenant "plato" --name "plato-resources"
    --limits "vm.memory 100 GB, vm.cpu 100 COUNT, vm 100 COUNT,
    persistent-disk 100 COUNT, persistent-disk.capacity 200 GB"
photon project create --tenant "plato" --name "plato-prjt"
    --resource-ticket "plato-resources"
    --limits "vm.memory 100 GB, vm.cpu 100 COUNT, vm 100 COUNT,
    persistent-disk 100 COUNT, persistent-disk.capacity 200 GB"
photon tenant set "plato"
photon project set "plato-prjt"
```

#### 6.1.5 Creating Resources for Use in the Cluster

The following command creates a flavor for a small VM that you'll use when you create the Kubernetes cluster.

```
photon flavor create --name cluster-small -k vm --cost "vm 1 COUNT, vm.cpu 1 COUNT, vm.memory 2 GB"
```

If you are setting up a Kubernetes cluster for production purposes in an environment without memory constraints, you can omit this command because Photon Controller includes the following default flavors for clusters:

- cluster-master-vm. Photon Controller supplies this flavor as the default for a Kubernetes master node.
- cluster-other-vm, for etcd nodes.
- cluster-vm-disk, the default ephemeral disk for a Kubernetes cluster.

When you create a Kubernetes cluster, Photon Controller taps the default flavors unless you specify other flavors.

#### 6.1.6 Creating Resources for Containerized Applications

The following sequence of commands provisions some resources for applications. You can skip these commands if you want, but the resources they create might come in handy later when you deploy applications. For more information, see [Flavors](#).

```
photon -n flavor create --name "vm-basic" --kind "vm"
    --cost "vm 1 COUNT, vm.cpu 2 COUNT, vm.memory 2 GB"
photon -n flavor create --name "disk-eph" --kind "ephemeral-disk"
    --cost "ephemeral-disk 1 COUNT"
photon -n flavor create --name "disk-persist" --kind "persistent-disk"
    --cost "persistent-disk 1 COUNT"
```

### 6.1.7 Setting Up a Network for Use with the Cluster

Finally, make a network for the cluster and set it as the default:

```
photon network create --name "vm-network" -portgroups "VM Network"
```

From the output of the `photon network create` command, note the network ID and then use it to set the default network in the following command:

```
photon network set-default <network_ID>
```

### 6.1.8 Creating the Kubernetes Cluster

You are now ready to create a Kubernetes cluster by running the following command. Replace the example IP addresses with those from your ESXi and network environments. The IP address for the `master-ip` option should contain the static IP address that you want to assign to the Kubernetes cluster. The `etcd` option should also contain a static IP address.

```
photon cluster create -n kube-socrates -k KUBERNETES --master-ip 203.0.113.208
--etcd1 203.0.113.209 --container-network 10.2.0.0/16 --dns 203.0.113.1
--gateway 203.0.113.253 --netmask 255.255.0.0 -c 1 --vm_flavor cluster-small
```

The `cluster create` command prompts you for several inputs. You can press `Enter` to accept the defaults and type `1` to for a worker node, or you can specify the options that you want.

The `photon cluster create --help` command shows descriptions of the options' values. Here's a truncated sample of its output:

<code>--name, -n</code>	Cluster name
<code>--type, -k</code>	Cluster type (accepted values are KUBERNETES, MESOS, or SWARM)
<code>--dns</code>	VM network DNS server IP address
<code>--gateway</code>	VM network gateway IP address
<code>--netmask</code>	VM network netmask
<code>--master-ip</code>	Kubernetes master IP address (required for Kubernetes clusters)
<code>--container-network</code>	CIDR representation of the container network, e.g. '10.2.0.0/16' (required for Kubernetes clusters)
<code>--etcd1</code>	Static IP address with which to create etcd node 1 (required for Kubernetes clusters)

### 6.1.9 Checking the Cluster's Resources and Status

After provisioning the cluster and its resources, you can log in to the Photon Controller web interface to check them out:

```
http://<ip-address-of-load-balancer-or-mgt->
```

Take a few moments to click through the web UI. Examine the tenant who owns the project running Kubernetes:

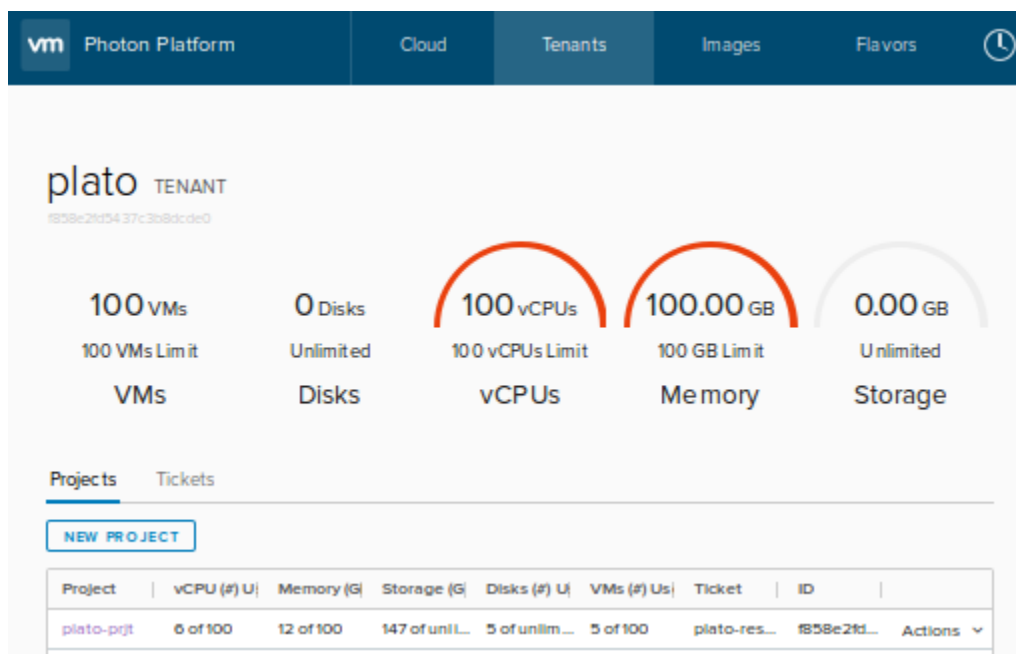


Figure 9: A Tenant in Photon Platform

In the list of the tenant's projects, you can click a project to view information about it:

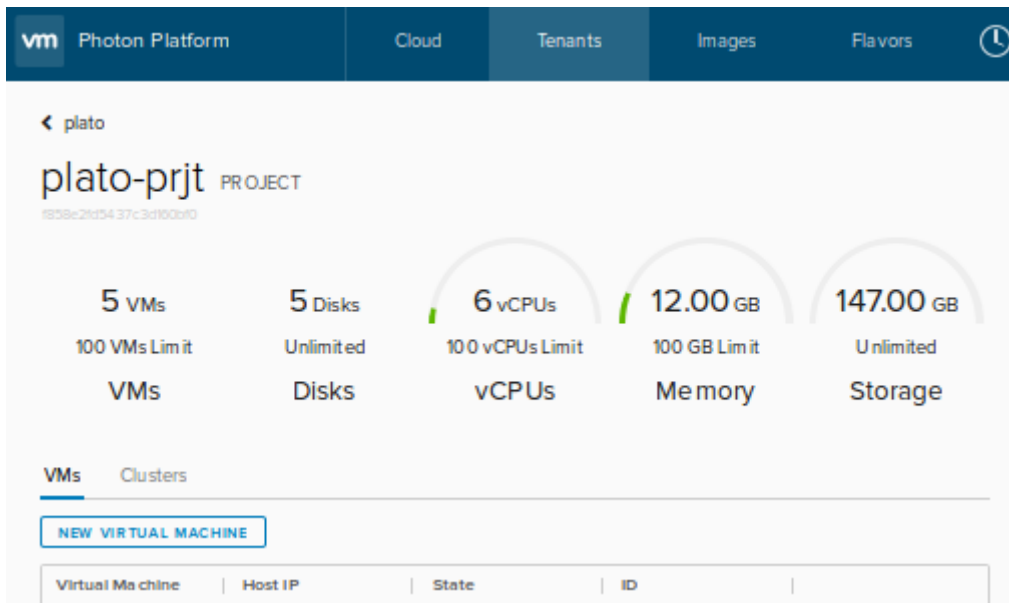


Figure 10: A Project in Photon Platform

And once you've reached the project, you can click **Clusters** and then click the name of a cluster to see its settings:

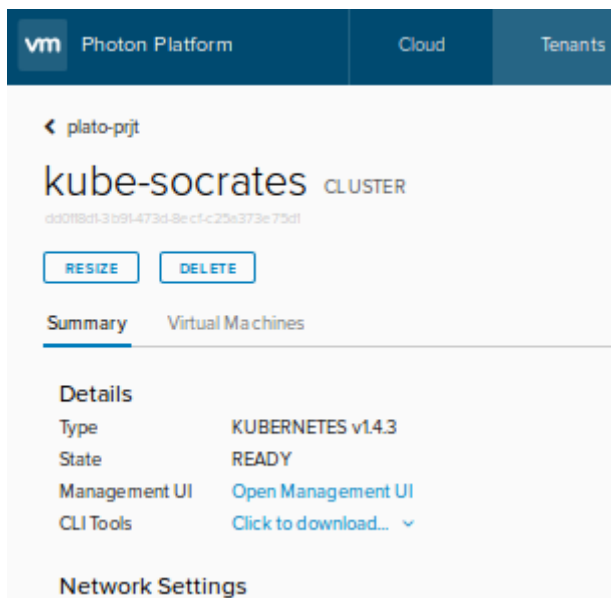


Figure 11: A Kubernetes Cluster



### 6.1.10 Opening the Kubernetes Dashboard

To open the Kubernetes web interface in a browser, go to the following URL:

`http://<ip-address-of-kubernetes-master>:8080`

Now you're ready to load and run an application in your Kubernetes cluster.

### 6.1.11 Deploying an nginx Web Server

Now that you've got a Kubernetes cluster up and running on Photon Controller, you can run a application as a service. This example deploys an nginx web server to demonstrate how to launch an application.

You'll need the Kubernetes command-line interface, `kubect1`, which you can quickly download through the Photon Platform web interface, as the following image illustrates:

vm

Photon Platform

Cloud

Tenants

< plato-prjt

kube-socrates

CLUSTER

dd0118d1-3b91-473d-8ecf-c25a373e75d1

RESIZE

DELETE

Summary

Virtual Machines

Details

TypeKUBERNETES v1.4.3

StateREADY

Management[Open Management UI](#)

UI

CLI Tools

Click to download... ▾

Linux (64-bit)

Linux (32-bit)

Windows (64-bit)

Windows (32-bit)

Mac OS (64-bit)

Network S

Gateway

Netmask

DNS

Container

network

Figure 12: Install kubectl

Or you can download the version of `kubectl` for a 64-bit Linux machine from the following URL, or see the [Kubernetes user guide](#) for instructions on how to download versions for other operating systems:

<https://storage.googleapis.com/kubernetes-release/release/v1.4.3/bin/linux/amd64/kubectl>

After you download `kubectl`, install it by changing its mode bits so that it is executable and then moving it to `/usr/local/bin`. Here's an example:

```
cd ~/Downloads/
chmod +x kubectl
sudo mv kubectl /usr/local/bin
```

Here's a quick trick to create a Kubernetes configuration file to simplify the commands that you will run:

```
kubectl config set-cluster default --server=http://<ip-address-of-kubernetes-master>:8080
kubectl config set-context default --cluster=default
kubectl config use-context default
```

Now you can quickly check your nodes' status, and you can omit the `server` option in the commands because the default is set:

```
kubectl get nodes
NAME                STATUS    AGE
198.51.36.6         Ready    25d
198.51.45.117      Ready    25d
```

You're now ready to deploy the nginx web server as a [service](#). Copy the following block of code into a file named `nginx.yml`. The code block contains the configuration for running the nginx web server on Kubernetes.

By setting the `type` field to `NodePort`, this YAML file instructs the Kubernetes master to allocate a port from the default node port range of 30000 to 32767. Each node proxies the same port to your service. A node port exposes a node's IP address; for more information, see the [Kubernetes documentation](#).

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-demo-service
  labels:
    app: nginx-demo
spec:
  type: NodePort
  ports:
  - port: 80
    protocol: TCP
    name: http
  selector:
    app: nginx-demo
---
```

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: nginx-demo
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx-demo
    spec:
```

```

containers:
- name: nginx-demo
  image: nginx
  ports:
  - containerPort: 80

```

Run the following command to create the nginx service on the Kubernetes cluster; you might need to modify the path the YAML file if it's not in your current working directory:

```
kubectl create -f nginx.yml
```

```

service "nginx-demo-service" created
replicationcontroller "nginx-demo" created

```

Now view the services running on the cluster:

```
kubectl get svc
```

NAME	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	10.0.0.1	<none>	443/TCP	3d
nginx-demo-service	10.0.0.202	<nodes>	80/TCP	5h

Finally, run the following command to obtain the NodePort on which the nginx web service is exposed as a service:

```
kubectl describe svc nginx-demo-service
```

```

Name:          nginx-demo-service
Namespace:     default
Labels:        app=nginx-demo
Selector:      app=nginx-demo
Type:          NodePort
IP:            10.0.0.202
Port:          http      80/TCP
NodePort:      http      30786/TCP
Endpoints:     10.2.100.7:80,10.2.100.8:80,10.2.100.9:80 + 3 more...
Session Affinity:  None

```

You can also see the nginx workload running in the Kubernetes web interface:

kubernetes

Workloads

+ CREATE

Admin

Namespaces

Nodes

Persistent Volumes

Namespace

default

Workloads

Deployments

Replica Sets

Replication Controllers

Replication controllers

Name	Labels	Pods	Age	Images
✓ nginx-demo	app: nginx-demo	3 / 3	16 minutes	nginx

Pods

Name	Status	Restarts	Age
✓ nginx-demo-5e35z	Running	0	16 minutes
✓ nginx-demo-hvhrv	Running	0	16 minutes
✓ nginx-demo-ri9d7	Running	0	16 minutes

Figure 13: Kubernetes Workloads

That's it. Now you can connect to the nginx web server by launching a web browser and pointing it at the IP address of the Kubernetes master plus the **NodePort** on which the service is running, which is **30786** in this case:

`http://203.0.113.208:30786`

You should see the nginx welcome screen:

# Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working.  
Further configuration is required.

For online documentation and support please refer to [nginx.org](http://nginx.org).  
Commercial support is available at [nginx.com](http://nginx.com).

*Thank you for using nginx.*

Figure 14: nginx Welcome Screen

### 6.1.12 Troubleshooting Cluster Creation

If the creation of the Kubernetes cluster failed, run the following photon command to clean up clusters that might be in an error state.

Keep in mind that this command deletes all your clusters, so you should run it only if you are trying to deploy your first one:

```
photon -n cluster list | awk '{print $1}' | xargs -n 1 photon -n cluster delete
```

Another troubleshooting method is to log into the VMs that Photon Controller creates for the Kubernetes master and the etcd nodes. The default password for the root account is `changeme`. You should change it the first time you log in.

After logging in, examine the system's status and logs as well as the Docker networking configuration.

You can get the ID of the cluster with the following the `photon cluster list` command; example:

```
photon cluster list
ID                               Name                Type           State  Worker Count
dd0118d1-3b91-473d-8ecf-c25a373e75d1 kube-socrates      KUBERNETES     READY  1
Total: 1
READY: 1
```

And then you can view more information about the cluster by running adding the ID to the `photon cluster show` command; example with abridged output:

```
photon -n cluster show dd0118d1-3b91-473d-8ecf-c25a373e75d1
dd0118d1-3b91-473d-8ecf-c25a373e75d1    kube-socrates      READY    KUBERNETES    1
dns:10.118.98.1
container_network:10.2.0.0/16
netmask:255.255.0.0
cluster_version:v1.4.3 g
ateway:10.118.101.253
cluster_ui_url:http://10.118.101.208:8080/ui
etcd_ips:10.118.101.209
master_ip:10.118.101.208
master-7a0c629b-52dc-49c9-b58f-f37100948487 10.118.101.208
etcd-96cae740-3c65-4ac8-96e9-c6b170990934 10.118.101.209
```

#### 6.1.12.1 Restoring Default Flavors

If you mistakenly deleted Photon's default flavors for Kubernetes clusters, you can restore them by running the following commands as the system administrator:

```
photon -n flavor create --name "cluster-master-vm" --kind "vm"
                                --cost "vm 1 COUNT, vm.cpu 4 COUNT, vm.memory 4 GB"
photon -n flavor create --name "cluster-other-vm" --kind "vm"
                                --cost "vm 1 COUNT, vm.cpu 1 COUNT, vm.memory 2 GB"
photon -n flavor create --name "cluster-vm-disk" --kind "ephemeral-disk"
                                --cost "ephemeral-disk 1 COUNT"
```

#### 6.1.12.2 Dealing with “Not Enough Memory Resource” Errors

The default flavors for a Kubernetes cluster show you how much memory a default Kubernetes cluster requires:

- A master node requires 4 CPUs and 4 GB of RAM.
- Each worker node that's running as its own VM takes 1 CPU and 2 GB of RAM.
- Each etcd node that's running as its own VM takes 1 CPU and 2 GB or RAM.

So, if you were to use the default flavors to build a Kubernetes cluster with a master, three workers, and three etcd nodes all running on their own VMs, the total memory requirement would be 10 CPUs and 16 GB of RAM.

If you're not tracking the memory usage of the ESXi machines in your Photon Controller, you might get an error code saying "NotEnoughMemoryResource." There are several ways to address the error:

- Reduce the size of the cluster you are trying to create.
- Free up resources by deleting unused virtual machines.
- Add additional memory or ESXi machines to the cluster.
- Delete unused Kubernetes clusters.

Another alternative for working in an environment with memory constraints is to make a small flavor and use it when you build a cluster, as this article did earlier when it demonstrated how to create a cluster.

#### 6.1.12.3 Deleting a Cluster to Reclaim Space

To delete a cluster to free up space for a new cluster in an environment with memory constraints, run this command:

```
photon cluster delete <cluster_ID>
```

#### 6.1.12.4 Deleting a Kubernetes Image

If you want to wipe out the Kubernetes image for some reason, such as recovering the storage space for an image that's old or no longer in use, you can delete it. You should, however, make sure that it's disabled as the cluster type before you delete it:

```
photon deployment disable-cluster-type -k KUBERNETES -i <Kubernetes_image_ID>
photon image delete <kubernetes_image_filename>
```

#### 6.1.13 Related

[Deploying Tomcat](#) on a Kubernetes cluster.

## 6.2 Obtaining and Uploading Images for Clusters

In a production cloud environment, a set of base images are made available for creating virtual machines. If you're wondering why you need base images, take a look at the [FAQ](#).

If you're working with Photon Controller in Workstation or on ESXi, you must download the images. On Fusion, the base images are automatically downloaded for you if you're installing Photon Controller by using the installation script.

### 6.2.1 Downloading the Base Image for a Cluster

You can download the base image from the following location:

- [Kubernetes](#)



### 6.2.2 Uploading Images to Photon Controller

Once you've download the images, you can upload them to Photon Controller. For more information, see [Images](#). Here's how to upload a base image for a cluster:

```
photon image create <kubernetes_image_filename>
-n photon-kubernetes-vm.vmdk -i EAGER
```

To save space, upload only the images for the orchestration framework that you want to work with.

The upload takes some time, and each image consumes some disk space.

### 6.2.3 Enabling The Cluster Type for an Image

Get the image ID for your uploaded cluster image:

```
photon image list
```

Get the deployment ID:

```
photon deployment list
```

Enable each cluster type with the its image ID:

```
photon deployment enable-cluster-type "deployment_ID"
-k KUBERNETES -i "Kubernetes Image ID"
```

Once you've uploaded the base images and enabled the cluster types for your images, you can begin working directly with a cluster framework. See [Creating a Kubernetes Cluster](#).

## 6.3 Running Tomcat on a Kubernetes Cluster

Here's how to set up and run an Apache Tomcat server on a Kubernetes cluster. The following examples assume that you have prepared Photon Controller to deploy Kubernetes clusters by following the instructions in [Creating a Kubernetes Cluster](#).

### 6.3.1 Spinning Up a Cluster for Tomcat

First, create a new cluster on Photon Controller for the Tomcat server:

```
photon cluster create -n Kube2 -k KUBERNETES --dns 192.168.209.2 --gateway 192.168.209.2
--netmask 255.255.255.0 --master-ip 192.168.209.35 --container-network 10.2.0.0/16
--etcd1 192.168.209.36 -c 2
```

The container network should be in CIDR format (for example, 10.2.0.0/16).

If you're deploying the cluster on VMware Workstation, skip the creation of etcd node 2 to work within the size constraints of your environment.

### 6.3.2 Deploying an App to the Cluster

Now we want to deploy a basic web server application onto the Kubernetes cluster we created. Make sure you have the `kubectl` tool installed on your workstation.

### 6.3.2.1 Linux, Unix, and Mac Users

- Download the `kubectl` utility for Mac from <https://storage.googleapis.com/kubernetes-release/release/v1.4.3/bin/darwin/amd64/kubectl>
- Download the `kubectl` utility for Linux from <https://storage.googleapis.com/kubernetes-release/release/v1.4.3/bin/linux/amd64/kubectl>
- Ensure the utility is executable: `chmod u+x kubectl`
- Put `kubectl` in your path (optional for Mac users): `mv kubectl /usr/local/bin`

### 6.3.2.2 Windows Users

- Download the `kubectl` utility for Microsoft Windows from Google [here](#)
- Unzip and untar the downloaded zip and use `kubectl.exe` from the source tree as follows:

`platforms/windows/amd64/kubectl.exe`

### 6.3.2.3 Download Replication Controller and Service Files

Grab the files below:

- Apache Tomcat [Replication Controller yml](#)
- Apache Tomcat [Service yml](#)

Now we can create an application by deploying a replication controller and the corresponding service. There may be a several minute delay while the image downloads on the first deployment.

- `kubectl -s 192.168.209.35:8080 create -f <path_to_rc>.yml`
- `kubectl -s 192.168.209.35:8080 create -f <path_to_service>.yml`

The pods should now be running. Confirm with this command:

```
kubectl -s 192.168.209.35:8080 get pods
```

You should see “Running” on the pods you’ve created.

Congrats: You now have an Apache Tomcat server running on the Kubernetes cluster.

### 6.3.2.4 Check Out the Application

We can view our Tomcat application at the following URL:

`http://192.168.209.35:30001`

## 6.3.3 Scaling with Kubernetes

We can scale out to multiple Replication Controllers quite easily by using `kubectl`:

```
kubectl -s 192.168.209.35:8080 scale --replicas=2 rc tomcat-server
```

After scaling, use `kubectl` to make sure you have two copies of the Tomcat Server pod:

```
kubectl -s 192.168.209.35:8080 get pods
```

## 6.3.4 Dealing with Environment Size Limitations

After you deploy your app on this cluster, you may need to delete it in order to create additional clusters in a small environment. When you are ready to delete it, run the following command:

```
photon cluster delete <cluster_uuid>
```

## 6.4 Managing Clusters

There are numerous helpful commands that can be used to interact with clusters created on Photon Controller.

- List cluster: `photon cluster list`
- View cluster details: `photon cluster show <cluster_uuid>`
- Show cluster VMs: `photon cluster list_vms <cluster_uuid>`
- Deleting cluster: `photon cluster delete <cluster_uuid>`

For additional commands and more information, run the following command:

```
photon cluster --help
```

You can also view the help for the subcommands; example:

```
photon cluster show -h
```

Here's the output of `photon cluster --help`:

```
photon cluster --help
NAME:
    photon cluster - Options for clusters
USAGE:
    photon cluster command [command options] [arguments...]
COMMANDS:
    create          Create a new cluster
    show            Show information about a cluster.
    list            List clusters
    list_vms        List the VMs associated with a cluster
    resize          Resize a cluster
    delete          Delete a cluster
    trigger-maintenance Start a background process
                    to recreate failed VMs in a cluster
    cert-to-file    Save the CA Certificate to a file with the
                    specified path if the certificate exists
    help, h         Shows a list of commands or help for one command
OPTIONS:
    --help, -h    show help
```

## 7 Information for Developers

### 7.1 Setting Up Your Own Load Balancer

If you do not use Photon Controller's load balancer (HAProxy), you might want to set up your own. Here is some information for you.

#### 7.1.1 Ports

**Port 9000:** Photon Controller uses Port 9000 for unauthenticated connections to the RESTful API.

**Port 80:** If Photon Controller is set up without using Lightwave authentication, Photon Controller uses Port 80 for HTTP connections to the web interface. It is not recommended to use Photon Controller without authentication.

**Port 4343:** Photon Controller uses Port 4343 for HTTPS connections to the web interface when Photon Controller is set up to use Lightwave authentication.

**Port 443:** Photon Controller uses Port 443 for command-line and API HTTPS connections when Photon Controller is set up to use Lightwave authentication.

### 7.1.2 Headers

**Host:** The `Host` header is used for constructing URIs in the REST API to ensure the URIs match the incoming host name or IP address.

**X-Forwarded-Proto:** The `X-Forwarded-Proto` header is used for constructing URIs in the REST API to ensure the URIs match the incoming protocol.

## 7.2 Compiling the Command-Line Utility

The easiest way to get started with the Photon Controller CLI client is to [download the pre-compiled version](#). Although you can compile it from the source code if you want, there are a limited number of reasons you would want to go about compiling the binary:

- You like compiling things from source. No problem, we're like that, too.
- You are a developer that's contributing to Photon Controller and added functionality to the CLI that you want to test.
- You've hit a bug and are compiling a newer version with the fix. Please reach out to us, though. We'd be happy to throw the "fixed" compiled binary up on a CDN so that other users don't have to go through the trouble of compiling this by hand if they don't want to.

Still want to compile? Keep reading.

### 7.2.1 Compiling CLI from source

It's best to follow the instructions on the [photon-controller-cli](#) repository; the directions on the Photon Controller CLI repository strictly take precedence over the instructions here. But here's a quick rundown of how to build the CLI tools:

- Ensure you have Go [installed](#)
- Download and build the Photon CLI: `go get github.com/vmware/photon-controller-cli/photon`

If you need to build the CLI against a specific commit you'll have to follow the instructions on the [photon-controller-cli repo](#).

### 7.2.2 Compatibility

When we spin a release, the accompanying CLI tools will also be released in binary format. There might be changes that break compatibility, so it's best to use the binary we release or ensure that the commit you compile against is compatible.

### 7.2.3 Testing Binary

You can test the binary by running it without any parameters:

```
% bin/photon
```

You should then see the `photon help` output:

NAME:  
 photon - Command line interface for Photon Controller

USAGE:  
 photon [global options] command [command options] [arguments...]

VERSION:  
 Git commit hash: 98db5e4

COMMANDS:

auth	options for auth
system	options for system operations
target	options for target
tenant	options for tenant
host	options for host
deployment	options for deployment
resource-ticket	options for resource-ticket
image	options for image
task	options for task
flavor	options for flavor
project	options for project
disk	options for disk
vm	options for vm
network	options for network
cluster	Options for clusters
availability-zone	options for availability-zone
help, h	Shows a list of commands or help for one command

GLOBAL OPTIONS:

--non-interactive, -n	trigger for non-interactive mode (scripting)
--log-file, -l	write logging information into a logfile at the specified path
--output, -o	select output format
--detail, -d	print the current target, user, tenant and project
--help, -h	show help
--version, -v	print the version

#### 7.2.4 Hitting a Problem?

Chances are there's something wrong with the way your Go environment was set up. If you're really stuck, ping us on GitHub.

### 7.3 Installing Photon Controller on Photon OS

Photon Controller can be installed using RPMs on Photon OS. To pull latest RPMs and install using TDNF command line tool on Photon OS, you need to create a file `/etc/yum.repos.d/photon-controller.repo` and add the following content into it.

```
[photon-controller]
name=VMware Photon Controller Core Services
baseurl=https://dl.bintray.com/vmware/photon-controller/photonos1.0/v1.0.0
gpgkey=file:///etc/pki/rpm-gpg/VMWARE-RPM-GPG-KEY
gpgcheck=0
enabled=1
```

```
skip_if_unavailable=True
```

After the file is created, run the following command to refresh the tdnf repo cache.

```
tdnf makecache
```

And now you can install Photon Controller using the following command:

```
tdnf install photon-controller
```

### Trying bleeding edge development bits

If you want to try out the latest code from the development branch, you need to replace the baseurl in the repo file with either one of the following URLs. The stable development branch is at this URL:

```
baseurl=https://dl.bintray.com/vmware/photon-controller/photonos1.0/stable
```

The latest development branch, which might be unstable at times, is at this URL:

```
baseurl=https://dl.bintray.com/vmware/photon-controller/photonos1.0/develop
```

## 8 Integrating VMware vSAN with Photon Platform

### 8.1 Setting Up VMware vSAN

Combining VMware [vSAN](#) with Photon Controller, Lightwave, and ESXi creates a powerful platform for cloud-native applications. vSAN establishes a software-defined storage cluster that transforms the local physical resources of ESXi hosts into a virtual pool of storage for Photon Controller.

After vSAN is installed and connected to Lightwave, Photon Controller can store images and other resources on the virtual storage cluster. And the virtual machines, Kubernetes clusters, and containerized applications running in Photon Controller can be assigned storage resources from the virtual storage cluster.

To set up vSAN for Photon Controller, you install the vSAN OVF on an ESXi host to create a vSAN management VM and connect it to the Lightwave server. You then connect to the vSAN management VM with SSH and run `rvc` commands to create a virtual storage area network.

#### 8.1.1 Prerequisites

- Photon Controller and Lightwave are installed.
- Lightwave is set up for Photon Controller, and you have created a domain and administrator group on the Lightwave server. See [Setting Up Authentication](#).
- The ESXi hosts must have Patch 201611001 (ESXi600-201611001), which you can find by searching for patches for ESXi 6.0.0 on the [My VMware Product Patches web site](#) at <https://my.vmware.com/group/vmware/patch>.
- You are using OVF Tool version 4.0 or later. To download the OVF Tool or to read its guide, see the [OVF Tool documentation](#).

#### 8.1.2 Requirements

Before you set up vSAN for Photon Controller, make sure that your environment meets the vSAN requirements. In general, vSAN requires at least three ESXi hosts, all with SSDs. See [vSAN Requirements](#).

All capacity devices, drivers, and firmware versions in your Virtual SAN configuration must be certified and listed in the Virtual SAN section of the VMware Compatibility Guide the following URL:

<http://www.vmware.com/resources/compatibility/search.php?deviceCategory=vsan>

There are also specific requirements for [hardware](#) and [networking](#).

You must also have the right vSAN license for your environment; see <http://www.vmware.com/products/virtual-san.html>.

For more information about installing and managing vSAN, see the [vSAN documentation](#).

## 8.2 Installing the vSAN Management Service

To download the vSAN OVA, see [VMware.com](#).

You deploy vSAN by using the OVF Tool to install the vSAN OVA on an ESXi host. From a workstation with the OVF Tool installed, execute the following command, replacing the values between < . . . > with the information from your environment and adding single quotes if your information contains spaces or special characters. Before you install vSAN, make sure that you have read the EULA for vSAN on VMware.com.

```
ovftool --acceptAllEulas --noSSLVerify --skipManifestCheck
--X:injectOvfEnv --overwrite --powerOffTarget --powerOn
--diskMode=thin
--X:apiVersion=5.5
--net:"NAT"=<network name, e.g. "VM VLAN">
--datastore=<datastore name, e.g. datastore1>
--name=VSAN-Management-VM
--prop:ip0=<VM IP address, e.g. 198.51.100.200>
--prop:netmask0=<netmask, e.g. 255.255.248.0>
--prop:gateway=<gateway address, e.g. 198.51.100.253>
--prop:hostname=<host name, e.g. vsan-mgmt-srv-vm>
--prop:root_password=<password for "root" user, default: "changeme". Change after login>
--prop:lw_hostname=<ip address of the Lightwave server, e.g. 198.51.100.124>
--prop:lw_domain=<Lightwave domain to join this VM to>
--prop:lw_password=<Lightwave administrator password>
--prop:lw_admingroup=<An Administrator group that is pre-created in Lightwave that has
access to lw_domain above. Default:ESXCloudAdmins >
    <path to OVA>/vsan.ova
vi://root:<host password here>@<host ip, e.g. 198.51.100.98>
```

### 8.2.1 Enabling SSL for vSAN

To secure the vSAN Management Service and enable SSL and authentication, you must provide these values:

- hostname
- lw\_hostname
- lw\_domain
- lw\_password
- lw\_admingroup

If these parameters are not supplied, the vSAN Management Service will still work but will not use SSL. Using the vSAN Management Service without SSL should be used only for testing purposes.

## 8.3 Prepare ESXi Hosts for vSAN

Connect to each ESXi host and run the following command to prepare it to be part of the vSAN cluster:

```
esxcli vsan network ipv4 add -i vmk0
```

## 8.4 Setting Up vSAN for Photon Controller

This section briefly takes you through the steps of setting up vSAN for Photon Controller. With the exception of including the Lightwave user and password in the commands, the commands are not specific to Photon Controller. You might have to alter certain arguments or variables to match your environment.

For more information about the commands and their options, see the [vSAN documentation](#) and the [VMware Ruby vSphere Console Command Reference for Virtual SAN](#).

To run the commands, connect to the VM on which you deployed the vSAN OVA by using SSH. When you connect, use the OVA root password that you set in the OVF Tool's parameters.

The following commands use an example Lightwave account, domain, and other information. Replace the examples with your own information. The examples use following values:

- Lightwave user account: pc-admin
- password: yoursecret
- Lightwave domain: esxcloud
- vSAN service port: 8006
- ESXi HOST\_IPS=(198.51.100.156 198.51.100.157 198.51.100.158 198.51.100.159 198.51.100.160 198.51.100.90) HOST\_USER on vSAN Management VM=root HOST\_PASSWORD on vSAN Management VM='secretpw\$01' CLUSTER\_NAME=vsan-cluster

In the commands, 127.0.0.1 is the local host's IP address.

### 8.4.1 Connect to the vSAN Management VM

First, connect to the vSAN Management VM that you deployed with the OVA:

```
ssh root@<IP address of vSAN Management VM>
```

Second, enter the Ruby vSphere Console by using the credentials of your Lightwave administrator's account; example:

```
rvc 'pc-admin@esxcloud':yoursecret@127.0.0.1:8006
```

### 8.4.2 Create a vSAN cluster

In the RVC console, run this command:

```
cluster.create 127.0.0.1/Global/computers/vsan-cluster
```

### 8.4.3 Join the ESXi Hosts to the vSAN Cluster

For each ESXi host that you want to add to the virtual storage network, run the following commands, replacing the user name and password with the user name and password of the ESXi host that you are adding by its IP address.

```
cd 127.0.0.1/Global/computers/  
ls  
cluster.add_host 0 <ip_address-of-esxi-host> -u "root" -p "secretpw$01"
```

In the current directory, the `ls` command shows the shortcut name for `vsan-cluster`. In this case, the shortcut name is 0; you can use it in the `cluster.add_host` command to indicate the cluster to which you want to add a host. The `ls` command is commonly used to reveal the shortcut name for a cluster or a host, depending on the current working directory in which the `ls` command is executed.



If you have many hosts that you want to include in the virtual SAN, you could use the `sshpass` utility along with a shell script to add the hosts. Here's an example. The script assumes that the user name and password are the same on every ESXi host that you specify by IP address for the `HOST_IPS`.

```
#!/bin/bash -xe
OVA_USER=root
OVA_PASSWORD=changeme
VSAN_SERVICE_PORT=8006
HOST_IPS=(198.51.100.156 198.51.100.157 198.51.100.158 198.51.100.159 198.51.100.160 198.51.100.90)
ESXi_HOST_USER=root
ESXi_HOST_PASSWORD='secretpw01'
CLUSTER_NAME=vsan-cluster

for i in ${HOST_IPS[@]};do
    sshpass -p "$OVA_PASSWORD" ssh -o StrictHostKeyChecking=no
        -o UserKnownHostsFile=/dev/null $OVA_USER@$OVA_IP
    "rvc '$LW_USER@$LW_DOMAIN':$LW_PASSWORD@127.0.0.1:$VSAN_SERVICE_PORT
    -c 'cd 127.0.0.1/Global/computers/'
        -c 'ls' -c 'cluster.add_host 0 $i -u "$HOST_USER"
        -p "$HOST_PASSWORD"' -c 'quit'"
done
```

#### 8.4.4 Enable auto claim disks on a host

```
cd 127.0.0.1/Global/computers/
ls
vsan.enable_vsan_on_cluster 0
```

#### 8.4.5 Auto claim disks on a host

```
cd 127.0.0.1/Global/computers/
ls
vsan.host_consume_disks 0
```

#### 8.4.6 Check the vSAN Cluster

```
vsan.cluster_info 0
vsan.disks_stats vsan-cluster
vsan.health.health_summary vsan-cluster
```

### 8.5 Detecting the vSAN Datastore After Setup

Once vSAN is installed and set up, Photon Controller detects the vSAN datastore after 15 minutes.

### 8.6 Removing the vSAN Datastore

If you want to permanently remove the vSAN datastore from the ESXi hosts, you must log in to each ESXi host and run the following command:

```
esxcli vsan cluster leave
```

For more information, see the vSAN documentation.

## 8.7 Scripting the Deployment

This article hints that you could create a shell script that when you run it with the `sshpass` utility, it could install vSAN for you. Here's an example. You would, of course, have to set the some of the values as environmental variables and change other values set in the script to match those of your environment. The script assumes that the user name and password of all the ESXi hosts in the cluster are exactly the same.

```
#!/bin/bash -xe

if [ -z "$OVA_IP" ]; then
    echo "OVA_IP needs to be set"
    exit -1
fi
if [ -z "$OVA_USER" ]; then
    echo "OVA_USER needs to be set"
    exit -1
fi
if [ -z "$OVA_PASSWORD" ]; then
    echo "OVA_PASSWORD needs to be set"
    exit -1
fi
if [ -z "$LW_USER" ]; then
    echo "LW_USER needs to be set"
    exit -1
fi
if [ -z "$LW_PASSWORD" ]; then
    echo "LW_PASSWORD needs to be set"
    exit -1
fi
if [ -z "$LW_DOMAIN" ]; then
    echo "LW_DOMAIN needs to be set"
    exit -1
fi

VSAN_SERVICE_PORT=8006
HOST_IPS=(198.51.100.156 198.51.100.157 198.51.100.158
          198.51.100.159 198.51.100.160 198.51.100.90)
HOST_USER=root
HOST_PASSWORD='changeme'
CLUSTER_NAME=vsan-cluster

# Create a VSAN cluster
sshpass -p "$OVA_PASSWORD" ssh -o StrictHostKeyChecking=no
-o UserKnownHostsFile=/dev/null $OVA_USER@$OVA_IP
"echo y | rvc '$LW_USER@$LW_DOMAIN':$LW_PASSWORD@127.0.0.1:$VSAN_SERVICE_PORT
-c 'cluster.create 127.0.0.1/Global/computers/$CLUSTER_NAME' -c 'quit'"

# Join hosts to VSAN cluster
for i in ${HOST_IPS[@]};do
    sshpass -p "$OVA_PASSWORD" ssh -o StrictHostKeyChecking=no
    -o UserKnownHostsFile=/dev/null $OVA_USER@$OVA_IP
    "rvc '$LW_USER@$LW_DOMAIN':$LW_PASSWORD@127.0.0.1:$VSAN_SERVICE_PORT
    -c 'cd 127.0.0.1/Global/computers/'
    -c 'ls' -c 'cluster.add_host 0 $i -u "$HOST_USER" -p "$HOST_PASSWORD"' -c 'quit'"
```

```
done

# Enable auto claim disks
sshpass -p "$OVA_PASSWORD" ssh -o StrictHostKeyChecking=no
-o UserKnownHostsFile=/dev/null $OVA_USER@$OVA_IP
"rvc '$LW_USER@$LW_DOMAIN':$LW_PASSWORD@127.0.0.1:$VSAN_SERVICE_PORT
-c 'cd 127.0.0.1/Global/computers/'
-c 'ls' -c 'vsan.enable_vsan_on_cluster 0' -c 'quit'"

# Auto claim disks
sshpass -p "$OVA_PASSWORD" ssh -o StrictHostKeyChecking=no
-o UserKnownHostsFile=/dev/null $OVA_USER@$OVA_IP
"rvc '$LW_USER@$LW_DOMAIN':$LW_PASSWORD@127.0.0.1:$VSAN_SERVICE_PORT
-c 'cd 127.0.0.1/Global/computers/'
-c 'ls' -c 'vsan.host_consume_disks 0' -c 'quit'"
```

## 9 API

### 9.1 Viewing the Interactive API Documentation


To view the full documentation for the Photon Controller API, go to one of the following URLs on your running Photon Controller instance. Replace <IP-address> with the IP address of your load balancer:

- Authenticated: <https://IP-address/api>
- Unauthenticated: <http://IP-address:9000/api>

For help finding the IP address of your load balancer, see [Connecting to the Load Balancer](#).

For guidance on how to work with the API, see [Using the API](#).

Here's what the interactive API documentation looks like after you connect to it by using one of the methods above:

 **apife**

Explore

# /auth

Show/Hide | List Operations | Expand Operations | Raw

GET

/auth

Returns Authentication/ Authorization Info

## Response Class

Model | Model Schema

**Auth {**  
  **endpoint** (*String, optional*): Url of the Authentication Server Endpoint,  
  **port** (*Integer, optional*): The Authentication Server Port,  
  **enabled** (*boolean*): Flag that indicates if auth is enabled or not  
**}**

## Response Content Type

application/json ▾

Try it out!

Figure 15: alt text

## 9.2 Using the API

This document describes how to use the Photon Controller API. It is a RESTful API.

Although the following examples use the cURL command-line tool, but you can use your favorite REST client. The example also use the [jq](#) command-line tool to neatly format the responses from cURL: The tool is not required, but it aids in understanding the response.

### 9.2.1 The API

You can find the full documentation for the API from a running Photon Controller. Point your web browser at one of the following URLs:

- Authenticated: <https://IP-ADDRESS/api>
- Unauthenticated: <http://IP-ADDRESS:9000/api>

Here are a couple of tips to help you navigate the API:

#### 9.2.1.1 Unauthenticated APIs

Two APIs are unauthenticated:

1. `/auth`: This allows you to get information about how to contact Lightwave to get tokens.
2. `/available`: This is meant for use by a load-balancer to determine whether the API is available.

#### 9.2.1.2 Authorization

Not all users can use all APIs. See [authentication](#) for information on what is available to different users.

#### 9.2.1.3 Tenant and Project scope

You may ask a question that seems simple: what API call lists all the VMs? Because Photon Controller is multi-tenant, there is no such API call. VMs are owned by a single tenant and contained in a project. Instead of querying for all VMs in the system, you can query for all the VMs in a single project.

Disks are similar and are scoped within a project. Hosts are scoped within a deployment.

### 9.2.2 API Port Numbers

We recommend accessing the API through a load-balancer. If you use the default HAProxy load balancer included with Photon Controller, the API may be on one of two ports:

**Port 443:** The port used when authentication is in place. Authentication is recommended for all deployments.

**Port 9000:** The port used when there is no authentication. Do not use this insecure method of connecting to the API in a production environment.

### 9.2.3 Authentication

#### 9.2.3.1 Lightwave

Photon Controller uses [Lightwave](#) for authentication; see the instructions on how to [configure Lightwave](#).

### 9.2.3.2 Tokens

Almost all API calls require authentication tokens. There are two ways to get the token:

1. Use the command-line client. We strongly recommend this method because the internal details of getting a token may change in the future.
2. Using the API. This is documented for completeness, but it is not recommended and subject to change. Use it at your own risk.

Using the API to obtain a token involves the following two-step process:

1. Use Photon Controller's `/auth` API to find out how to contact Lightwave. (This is one of two APIs that is unauthenticated.)
2. Present credentials to Lightwave's `/openidconnect/token` API to receive a token. In response you will receive an access token that you can present to the Photon Controller APIs. The token will expire after some time depending on your local policies. You will also receive a refresh token that will allow you to get more access tokens. The refresh token also expires, but it has a longer expiration time than the access token.

### 9.2.3.3 Getting a token via the command-line

You can use the `photon` command-line to create a new token. For example, assuming: you have a user named 'houdini' with password 'secret': (tokens have been trimmed for legibility)

```
% photon auth get-api-tokens -u 'houdini' -p 'secret'
```

Access Token:

eyJhb...pgwI

Refresh Token:

eyJhb...aVKE

### 9.2.3.4 Getting a token through the APIs

#### 9.2.3.4.1 Querying the `/auth` API to find Lightwave

To find the instance of Lightwave being used by Photon Controller, query the `/auth` API. This API does not require authentication because it is required in order to authenticate.

Example:

```
% curl -k -s https://192.9.2.54/auth | jq .
{
  "enabled": true,
  "endpoint": "192.9.2.42"
}
```

Note that the port number is not included. Normally Lightwave will be on the standard HTTPS port, 443.

#### 9.2.3.4.2 Querying Lightwave to get an access token

Please note that tokens are normally quite long, but we shorten them to make this text more clear. Also note that using a refresh token involves a slightly different process.

To get a token from Lightwave, you will do a POST to the `/openidconnect/token` API. The POST of the API will be a string that includes four things:

1. grant\_type: password
2. username
3. password
4. scope: Use the string “openid offline\_access rs\_esxcloud at\_groups”. Note that this is subject to change.

These are combined in a body in a way that looks similar to a URL with parameters. For example, if your username was ‘houdini’ and your password was ‘secret’, the body would look like:

```
grant_type=password&username=houdini&password=secret&scope=openid offline_access rs_esxcloud at_groups'
```

Putting this into a POST request would look like this:

```
curl -k -s -X POST -H "Content-Type: application/x-www-form-urlencoded"
  -d 'grant_type=password&username=ec-admin@esxcloud&password=Passw0rd!&scope=openid
  offline_access rs_esxcloud at_groups' https://192.9.2.42/openidconnect/token | jq .
{
  "access_token": "eyJhbGc...NFD8k",
  "refresh_token": "eyJhbGc...uon-Q",
  "id_token": "eyJhbGc...0hyuo",
  "token_type": "Bearer",
  "expires_in": 7200
}
```

### 9.2.3.5 Using the token

Pass the token in the Authorization header, like this:

```
curl -s -k -H "Authorization: Bearer eyJhbGc...NFD8k" https://192.9.2.54/status | jq
{
  "components": [
    {
      "component": "HOUSEKEEPER",
      "status": "READY",
      ... output trimmed ...
    }
  ]
  "status": "READY"
}
```

## 10 Troubleshooting and Maintenance

### 10.1 Maintaining Hosts

Here’s how to perform maintenance on an ESXi host in a Photon Controller cluster by using the Photon command-line interface.

If the maintenance could affect the version of software, you should first remove the host from the Photon Controller cluster with **photon host delete**. After the maintenance is complete, you can add the host back into the cluster, which reinstalls the Photon Controller agent on the host.

Photon Controller holds an ESXi host in a state, usually a **ready** state. To perform maintenance, you must first place an ESXi host in a suspended state, remove all the VMs on it, and then place the host into a maintenance state.

### 10.1.1 Host States

1. **CREATING** Host is being created.
2. **NOT\_PROVISIONED** Host added, but no agent installed yet
3. **READY** Host is ready.
4. **MAINTENANCE** Host is in maintenance. No VM is running on the host.
5. **SUSPENDED** Host suspended. Existing VMs are still running but won't allow to add new VMs.
6. **ERROR** Host Error states. Such as provisioning failed, etc.
7. **DELETED** Infrastructure use only.

### 10.1.2 Suspending the Host and Entering Maintenance Mode

First, suspend the host. You can only suspend a host if it's in the **READY** state, and you can enter maintenance mode for a host only if it's in the suspended state and there are no VMs on it. Here's an example of how to suspend a host:

```
photon host suspend 40778c18-570b-479e-a92e-5801395376b9
SUSPEND_HOST completed for host entity 40778c18-570b-479e-a92e-5801395376b9
```

Second, remove all the VMs from the host.

```
photon host list-vms 40778c18-570b-479e-a92e-5801395376b9
ID                                     Name  State
4f2fe2d3-7c2a-4f35-b70a-a1d6fec90600  vm-1  READY
```

```
photon vm delete 4f2fe2d3-7c2a-4f35-b70a-a1d6fec90600
DELETE_VM completed for 'vm' entity 4f2fe2d3-7c2a-4f35-b70a-a1d6fec90600
```

Third, place the host in maintenance mode. A host cannot be put into maintenance mode if it contains VMs.

```
photon host enter-maintenance 40778c18-570b-479e-a92e-5801395376b9
ENTER_MAINTENANCE_MODE completed for host entity 40778c18-570b-479e-a92e-5801395376b9
```

Fourth, perform your maintenance and then exit maintenance mode:

```
photon host exit-maintenance 40778c18-570b-479e-a92e-5801395376b9
EXIT_MAINTENANCE_MODE completed for host entity 40778c18-570b-479e-a92e-5801395376b9
```

## 10.2 Troubleshooting Installation and Operations

This document describes troubleshooting steps you can take if you run into issues installing or using Photon Platform. If you encounter a problem that's not addressed in this document, in the quick start guide, or in the documentation on the GitHub wiki, feel free to visit us on [Google Groups](#) or, if you prefer, open a GitHub [issue](#).

### 10.2.1 General Troubleshooting

#### 10.2.1.1 Log Files

The log files are located in the `/vagrant/log` directory in the Photon Controller VM. You can ssh into it by using the `root` and `vmware` default credentials.

The following logs are the most valuable:

- **deployer.log** – handles the cluster logging commands. If you have an error in a cluster create, for example, you might start here.



- **management-api.log** – captures the results of API calls. It's where you can look for errors that come back to your CLI client.

### 10.2.1.2 Image Upload Failure

Verify the file path you entered.

### 10.2.1.3 Reserve\_Resource Error

A Reserve\_Resource error may look something like this:

```
2015/11/10 01:08:12 esxcloud: Task '48b474d6-1773-4dca-a5cb-7a45a38e1904' is in error
state. Examine task for full details.API Errors: [esxcloud: { HTTP status: '0', code:
'InternalServerError', message: 'Failed to roll out SwarmEtcd. Error: MultiException
[java.lang.IllegalStateException: VmProvisionTaskService failed with error
[Task "CREATE_VM": step "RESERVE_RESOURCE" failed with error code "InternalServerError",
message "null"]. /photon/clustermanager/vm-provision-tasks/
24537ef5-5757-42ea-81ff-3b63cc5734b6]', data: 'map[]' }]
```

This error signifies that the scheduler was unable to place the VM on your ESXi host.

There can be several root causes:

- Your ESXi host may be low on resources (usually RAM if installing on Fusion or Workstation)
- You have not correctly added the host to Photon Controller. Carefully review the installation instructions.
- Confirm the host is in a **READY** state by running `photon host list`
- It is also possible one or more critical components in Photon Controller has stopped. All the components that make up Photon Controller run inside Docker containers

Here's how to check on the status of the containers:

- Log into 192.168.209.29 (`root/vmware`)
- Execute `docker ps`
- You should see 8 containers with status **up**
- If one is not running, check the Docker logs for further information

### 10.2.1.4 Image Upload Fails

Image uploads can fail if you have previously run the `photon host create` command and then deployed a new Photon Controller VM without first removing the agent VIB from the ESXi VM.

The original agent will attempt to register itself with your new Photon Controller using the old deployment configuration.

You can remove the VIB following [these](#) directions.

### 10.2.1.5 My services aren't registering properly

Upon deployment all control plane services should be registered in Zookeeper. You can confirm as follows:

```
zlookup -z zookeeper-hostIP:2181
```

## 10.2.2 Fusion and Workstation Troubleshooting

### 10.2.2.1 Confirm ESXi Deployed Correctly

You can verify that your ESXi node was deployed properly via `root@192.168.209.31` with password `VMware1!`.

### 10.2.2.2 Failure in Allocating Resources

You may run into resource contention issues if you're installing on Fusion or Workstation.

If you attempt to create a cluster and see messages containing a failure in "Allocate Resources," you may have used all the resources available on your ESXi VM. You can confirm this by logging in to the Host Client at:

`https://192.168.209.31/ui/`

and sign in with `root/VMware1!`.

The host navigator will show you the utilized CPU, memory and storage. Because many users will install this on their laptop via Fusion or Workstation, the memory will likely be the point of contention. Because many VMs will allocate 512MB - 1GB of memory it becomes easy to see how memory can become an issue.

Installation to a laptop via Fusion or Workstation will generally support only a single cluster running at one time. Mesos is particularly problematic because of the number of required components: Zookeeper, Master node, Marathon node, and finally worker nodes.

You *may* be able to shrink the instances that make up the cluster by changing flavors..

It's probably best to simply delete clusters if they're no longer being used, though.

### 10.2.2.3 SwarmMaster Rollout Failure

A user may occasionally hit a **Failed to rollout SwarmMaster** error, almost always in a Fusion or Workstation environment.

The symptoms surrounding this error are as follows:

- API call to Photon Controller to create cluster is sent
- System hangs at step 4/5 for more than 10 minutes
- Error is then reported

The cause for this error is networking configuration. If DHCP services are not enabled on VMNET\_8, the Controller cannot finish creating the cluster.

You can enable DHCP in `/Library/Preferences/VMware Fusion/networking` by setting `VMNET_8_DHCP` to `yes`.

It is also possible that memory is heavily constrained, which can cause Photon Controller containers to stop when contention is not alleviated. If you have enough RAM, increasing beyond the default 3GB on Fusion or Workstation deployments may help alleviate the problem.

You can also check *Out of Memory* errors by using `dmesg`. See the [Photon OS Linux Troubleshooting Guide](#).

### 10.2.2.4 Mesos Cluster Failure

The likely cause of a Mesos cluster failure is that the system has run out of available resources.

In a Fusion or Workstation installation, memory contention can be a problem. Mesos clusters will take up more RAM than Kubernetes and Swarm clusters.

There are some additional steps you can take to troubleshoot:

- Verify that you have at least 7GB of RAM available on your ESXi Host VM.
- Verify the network configuration for the Zookeeper nodes:
- IP addresses should be assigned in 192.168.209.3 - 127 range
- Gateway and netmask should be configured properly
- IP address is not already in use (i.e., no address collision)
- Verify that you have uploaded the Mesos image and the name starts with `photon-mesos-vm`

As mentioned above, you may be able to shrink the instances that make up the cluster by changing flavors; see the documentation on images and flavors.

It's probably best to delete clusters if they're no longer being used, though.

### 10.2.3 Uninstallation

#### 10.2.3.1 Removing Agent VIB

When a host is made available for **CLOUD** VMs via the `photon create host` command, the Controller will install a VIB on the ESXi host.

The VIB is a Controller Agent and will attempt to register itself to on the IP and port of the Control plane (in the case of Fusion / Workstation installations, 192.168.209.29).

If you remove the Controller VM and re-deploy it, you must also remove the VIB.

Failure to follow this procedure may present itself through random internal errors. Follow the steps below to remove the VIB from your ESXi hosts:

- SSH into the host: `ssh root@<esxi_host>`
- Determine if VIB is installed: `esxcli software vib list | grep photon-controller-agent`
- Remove the VIB: `esxcli software vib remove -n photon-controller-agent`

## 11 Appendix I: Command-Line Examples

### 11.0.4 Getting help

The photon CLI includes extensive usage description that you can use to discover various operations.

For instance, you can see the top-level commands with:

```
% photon help
```

```
NAME:
```

```
    photon - Command line interface for Photon Controller
```

```
USAGE:
```

```
    photon [global options] command [command options] [arguments...]
```

```
VERSION:
```

```
    Git commit hash: 4607b29
```

```
COMMANDS:
```

```
    auth      options for auth
    system    options for system operations
    target    options for target
    tenant    options for tenant
    host       options for host
    deployment options for deployment
    resource-ticket options for resource-ticket
    image     options for image
    task       options for task
    flavor     options for flavor
    project    options for project
    disk       options for disk
    vm         options for vm
```

```
network    options for network
cluster    Options for clusters
availability-zone options for availability-zone
help, h     Shows a list of commands or help for one command
```

GLOBAL OPTIONS:

```
--non-interactive, -n trigger for non-interactive mode (scripting)
--help, -h           show help
--version, -v        print the version
```

You can see help for an individual command too:

```
% photon tenant --help
```

NAME:

```
photon tenant - options for tenant
```

USAGE:

```
photon tenant command [command options] [arguments...]
```

COMMANDS:

```
create    Create a new tenant
delete    Delete a tenant
list      List tenants
set       Select tenant to work with
show      Show current tenant
tasks     Show tenant tasks
set_security_groups Set security groups for a tenant
help, h   Shows a list of commands or help for one command
```

OPTIONS:

```
--help, -h show help
```

### 11.0.5 Interactive vs. Non-interactive mode

All commands work in two mode: interactive and non-interactive. Interactive mode will prompt you for parameters you do not provide on the command-line and will print human-readable output. Non-interactive mode will not prompt you and will print machine-readable output.

### 11.0.6 IDs

Objects in Photon Controller are given unique IDs, and most commands refer to them using those IDs.

### 11.0.7 Setting a target

Before you can use the photon CLI, you need to tell it which Photon Controller to use.

Usage: `photon target set <PHOTON-CONTROLLER-URL>`

Example:

```
% photon target set https://198.51.100.41
API target set to 'https://198.51.100.41'
```

If you are not using HTTPS, specify the port:

```
% photon target set http://198.51.100.41:9000
API target set to 'http://198.51.100.41:9000'
```

### 11.0.8 Tenants

Creating a tenant will tell you the ID of the tenant:

Usage: `photon tenant create <TENANT-NAME>`

Example:

```
% photon -n tenant create cloud-dev
cloud-dev 502f9a79-96b6-451d-bfb9-6292ca5b6cfd
```

You can list all tenants:

```
% photon -n tenant list
502f9a79-96b6-451d-bfb9-6292ca5b6cfd  cloud-dev
```

### 11.0.9 Set tenant for other commands

Many commands take a `-tenant` parameter because the object in question is owned by a tenant. As a convenience, you can avoid passing that parameter, you can set the tenant for future commands:

Usage: `photon tenant set <TENANT-NAME>`

Example:

```
% photon tenant set cloud-dev
Tenant set to 'cloud-dev'
```

The tenant will be stored in a configuration file in your home directory, within a subdirectory named *.photon-config*.

You can see what the current tenant is:

```
% photon tenant get
Current tenant is 'cloud-dev' 502f9a79-96b6-451d-bfb9-6292ca5b6cfd
```

### 11.0.10 Resource tickets

You create resource tickets to control the allocations granted to projects, which are owned by tenants.

A resource ticket must specify the number of VMs that can be created as well as the total amount of RAM consumed by those VMs. It's possible to have user-defined resources as well. These are specified as comma-separated limits, and each limit is a set of three things:

- Name (e.g. `vm.memory`)
- Value (e.g. `2000`)
- Units (`GB`, `MB`, `KB`, `COUNT`)

Creating a ticket in the current tenant (see above, or use the `-tenant` flag):

Usage `photon resource-ticket create --name <RESOURCE-TICKET-NAME> --limits "<LIMITS>"`

Example:

```
% photon -n resource-ticket create --name cloud-dev-resources
--limits "vm.memory 2000 GB, vm 1000 COUNT"
32ad527e-d21a-4b2a-a235-b0883bd64354
```

Creating a ticket with user-defined resources:

```
% photon -n resource-ticket create --name cloud-dev-resources
--limits "vm.memory 2000 GB, vm 1000 COUNT vm.potrzebie 250 COUNT"
32ad527e-d21a-4b2a-a235-b0883bd64354
```

Viewing tickets:

```
% photon -n resource-ticket list
1
32ad527e-d21a-4b2a-a235-b0883bd64354
cloud-dev-resources vm.memory:2000:GB,vm:1000:COUNT
```

```
% photon -n resource-ticket show cloud-dev-resources
cloud-dev-resources 32ad527e-d21a-4b2a-a235-b0883bd64354
vm.memory:2000:GB,vm:1000:COUNT vm.memory:0:GB,vm:0:COUNT
```

```
% photon resource-ticket show cloud-dev-resources
```

ID	Name	Limit	Usage
32ad527e-d21a-4...	cloud-dev-resources	vm.memory 2000 GB vm 1000 COUNT	vm.memory 1000 GB vm 500 COUNT

### 11.0.11 Projects

A project is owned by a tenant and all VMs are created within a project. Each project is associated with a resource ticket that controls the total resources that can be used. See above for more information about resource tickets.

A project has a set of limits. These are specified just like the resource ticket above, but they must not exceed the limits in the associated resource ticket.

Creating a project:

Usage:

```
photon project create --resource-ticket <RESOURCE-TICKET-NAME> --name <PROJECT-NAME>
--limits <LIMITS>
```

```
% photon -n project create --resource-ticket cloud-dev-resources
--name cloud-dev-staging --limits "vm.memory 1000 GB, vm 500 COUNT"
fabb9236-d0a4-4d30-8935-ee65d6729f78
```

Viewing projects:

```
% photon -n project list
fabb9236-d0a4-4d30-8935-ee65d6729f78 cloud-dev-staging
vm.memory:1000:GB,vm:500:COUNT vm.memory:0:GB,vm:0:COUNT
```

Setting the project (applies to commands that require a project, like creating a VM). If you prefer, you can pass the `-project` flag:

```
% photon -n project set cloud-dev-staging
```

### 11.0.12 Flavors

When a VM is made, it is described using two kinds of flavors: VM and disk. The flavors describes how many resources are consumed by the VM from the resource ticket.

The cost argument specifies a set of costs, each separated by commas. Each cost consists of three value:

- Name
- Value, which will be subtracted from the resource ticket when the VM is created
- Units: GB, MB, KB, B, or COUNT

Note that VM flavors must specify at least the vm.cpu and vm.memory costs. Other user-defined costs may be included as well, if desired. They should match the resources in the resource ticket.

Creating a VM flavor with with 1 VM, 1 CPU and 2 GB RAM:

Usage: `photon flavor create --name <FLAVOR-NAME> --kind <KIND> --cost <COST>`

Example:

```
% photon -n flavor create --name "cloud-vm-small" --kind "vm"
--cost "vm 1.0 COUNT, vm.cpu 1.0 COUNT, vm.memory 2.0 GB"
ddfb5be0-3355-46d3-9f2f-e28750eb201b
```

Creating a VM flavor with user-defined attributes:

```
% photon -n flavor create --name "cloud-vm-small" --kind "vm"
--cost "vm 1.0 COUNT, vm.cpu 1.0 COUNT, vm.memory 2.0 GB vm.potrzebie 10"
ddfb5be0-3355-46d3-9f2f-e28750eb201b
```

Creating a disk flavor:

```
% photon -n flavor create --name "cloud-disk"
--kind "ephemeral-disk" --cost "ephemeral-disk 1.0 COUNT"
78efc53a-88ce-4f09-9b5d-49662d21e56c
```

Viewing flavors:

```
% photon -n flavor list
78efc53a-... cloud-disk ephemeral-disk ephemeral-disk:1:COUNT
ddfb5be0-... cloud-vm-small vm vm:1:COUNT,vm.cpu:1:COUNT,vm.memory:2:GB
```

```
% photon flavor show ddfb5be0-3355-46d3-9f2f-e28750eb201b
```

Flavor ID: ddfb5be0-3355-46d3-9f2f-e28750eb201b

Name: cloud-vm-small

Kind: vm

Cost: [vm 1 COUNT vm.cpu 1 COUNT vm.memory 2 GB]

State: READY

### 11.0.13 Images

Uploading an image (OVA, OVF, or VMDK). The replication type is either EAGER or ON\_DEMAND

Usage: `photon image create <IMAGE-FILENAME> -n <IMAGE-NAME> -i <TYPE>`

Example:

```
% photon image create photon.ova -n photon-os -i EAGER
Created image 'photon-os' ID: 8d0b9383-ff64-4112-85db-e8111e2269fc
```

Viewing images:

```
% photon image list
Name      State  Size(Byte)  Replication_type  ReplicationProgress  SeedingProgress
photon-os  READY  16777216146  EAGER             100.0%               100.0%
```

Total: 1

```
% photon image show 8d0b9383-ff64-4112-85db-e8111e2269fc
```

```
Image ID: 8d0b9383-ff64-4112-85db-e8111e2269fc
```

```
Name: photon-os
```

```
State: READY
```

```
Size: 16777216146 Byte(s)
```

```
Image Replication Type: EAGER
```

```
Settings:
```

```
scsi0.virtualDev : lsilogic
```

```
ethernet0.virtualDev : vmxnet3
```

Deleting images:

```
% photon image delete 8d0b9383-ff64-4112-85db-e8111e2269fc
```

```
Are you sure [y/n]? y
```

```
DELETE_IMAGE completed for 'image' entity 8d0b9383-ff64-4112-85db-e8111e2269fc
```

Note that if you delete an image that is being used by a VM, it will go into the PENDING\_DELETE state. It will be deleted once all VMs that are using it have also been deleted.

```
% photon image show 8d0b9383-ff64-4112-85db-e8111e2269fc
```

```
Image ID: 8d0b9383-ff64-4112-85db-e8111e2269fc
```

```
Name: kube
```

```
State: PENDING_DELETE
```

```
Size: 16777216146 Byte(s)
```

```
Image Replication Type: EAGER
```

```
Image Replication Progress: 100%
```

```
Image Seeding Progress: 100%
```

```
Settings:
```

#### 11.0.14 VMs

When you create a VM, you must specify both the VM and disk flavors. The disks parameter lists a set of disks, separated by commas. Each disk is described by three values:

- name
- flavor
- Either “boot=true” or a size in GB for the disk

```
Usage: photon -n vm create --name <VM-NAME> --image <IMAGE-ID> --flavor <VM-FLAVOR>  
--disk <DISK-DESCRIPTION>
```

```
% photon -n vm create --name vm-1  
--image 8d0b9383-ff64-4112-85db-e8111e2269fc  
--flavor cloud-vm-small --disks "disk-1 cloud-disk boot=true"  
86911d88-a037-4576-9649-4df579abb88c
```

Starting a VM:

```
% photon vm start 86911d88-a037-4576-9649-4df579abb88c
```

```
START_VM completed for 'vm' entity 86911d88-a037-4576-9649-4df579abb88c
```

Viewing VMs. Note that the IP address will only be shown in the VM tools are installed on the VM:

```
% photon vm list
```

```
Using target 'http://198.51.100.41:9000'
```

ID	Name	State
86911d88-a037-4576-9649-4df579abb88c	vm-1	STARTED

```
Total: 1
```



STARTED: 1

```
% photon vm show 86911d88-a037-4576-9649-4df579abb88c
```

Using target 'http://198.51.100.41:9000'

VM ID: 86911d88-a037-4576-9649-4df579abb88c

```
Name:          vm-1
State:         STARTED
Flavor:        cloud-vm-small
Source Image:  8d0b9383-ff64-4112-85db-e8111e2269fc
Host:          198.51.100.190
Datastore:     56d62db1-e77c3b0d-7ebe-005056a7d183
Metadata:      map[]
Disks:
  Disk 1:
    ID:         2000d3a5-aaba-40c1-b08e-ba8a70be6112
    Name:        disk-1
    Kind:        ephemeral-disk
    Flavor:      78efc53a-88ce-4f09-9b5d-49662d21e56c
    Capacity:    15
    Boot:        true
Networks: 1
  Name:         VM Network
  IP Address:
```

Note that when the VM is created, it consumes some of the resources allocated to the project, based on the definitions in the flavor:

```
% photon project list
```

Using target 'http://198.51.100.41:9000'

ID	Name	Limit	Usage
fabb...	cloud-dev-staging	vm.memory 1000 GB	vm.cpu 1 COUNT
		vm 500 COUNT	vm.memory 2 GB
		vm 1 COUNT	
		ephemeral-disk.capacity 15 GB	
		ephemeral-disk 1 COUNT	

Total projects: 1

### 11.0.15 Hosts

Adding an ESX host:

Usage: 'photon host create -u -p -i

-tag -d "

```
% photon -n host create -u root -p MY-PASSWORD
-i 198.51.100.139 --tag 'CLOUD' -d prod-deployment
3a159e73-854f-4598-937f-909d503b1dc6
```

Viewing hosts:

```
% photon deployment list-hosts prod-deployment
ID                               State IP                               Tags
3a159e73-854f-4598-937f-909d503b1dc6 READY 198.51.100.139 CLOUD
a5411f8c-84b6-4b58-9670-7728db7c4cac READY 198.51.100.190 CLOUD
```

Total: 2

## 12 Appendix II: Deployment Template

This page includes several YAML templates that you can modify to deploy Photon Controller. Additional templates and explanations of the fields appear in the [Photon Controller Quick Start Guide](#).

### 12.0.16 Base Template

Here is a minimal template to deploy Photon Controller:

```
hosts:
  - metadata:
      MANAGEMENT_DATASTORE: mgmt_datastore
      MANAGEMENT_PORTGROUP: mgmt_network
      MANAGEMENT_NETWORK_NETMASK: <mgmt_netmask>
      MANAGEMENT_NETWORK_DNS_SERVER: <mgmt_dns_server>
      MANAGEMENT_NETWORK_GATEWAY: <mgmt_gateway>
      MANAGEMENT_VM_IPS: <mgmt_vm_ip>
      address_ranges: <mgmt_hypervisor_range>
      username: <mgmt_hypervisor1_username>
      password: <mgmt_hypervisor1_password>
      usage_tags:
        - MGMT
  - address_ranges: 198.51.100.13-198.51.100.15
      username: <cloud_host_username>
      password: <cloud_host_password>
      usage_tags:
        - CLOUD
deployment:
  resume_system: true
  image_datastores:
    - nfs
  auth_enabled: false
  syslog_endpoint: <syslog_server_ip>
  ntp_endpoint: <ntp_server_ip>
  use_image_datastore_for_vms: true
  loadbalancer_enabled: true
```

Here is a table that describes the parameters that you can define in the *metadata* leaf:

Metadata	Description
MANAGEMENT_DATASTORE	Datastore where your management VMs will be stored
MANAGEMENT_PORTGROUP	Port group your management VMs should utilize
MANAGEMENT_NETWORK_NETMASK	Netmask on your ESXi network
MANAGEMENT_NETWORK_DNS_SERVER	DNS server configured on your ESXi network
MANAGEMENT_NETWORK_GATEWAY	Network gateway for your ESXi network
MANAGEMENT_VM_IPS	IP addresses for the management VMs

### 12.0.17 Hosts on Disparate Networks

If your management VMs are located on different networks, you must create multiple host entries in your deployment yml file. An example is provided below.

```
- metadata:
  MANAGEMENT_DATASTORE: mgmt_datastore
  MANAGEMENT_PORTGROUP: mgmt_network1
  MANAGEMENT_NETWORK_NETMASK: <mgmt_netmas1k>
  MANAGEMENT_NETWORK_DNS_SERVER: <mgmt_dns_server>
  MANAGEMENT_NETWORK_GATEWAY: <mgmt_gateway1>
  MANAGEMENT_VM_IPS: <mgmt_vm1_ip>
  address_ranges: <mgmt_hypervisor_range>
  username: <mgmt_hypervisor1_username>
  password: <mgmt_hypervisor1_password>
  usage_tags:
    - MGMT
- metadata:
  MANAGEMENT_DATASTORE: mgmt_datastore
  MANAGEMENT_PORTGROUP: mgmt_network2
  MANAGEMENT_NETWORK_NETMASK: <mgmt_netmask2>
  MANAGEMENT_NETWORK_DNS_SERVER: <mgmt_dns_server>
  MANAGEMENT_NETWORK_GATEWAY: <mgmt_gateway2>
  MANAGEMENT_VM_IPS: <mgmt_vm2_ip>
  address_ranges: <mgmt_hypervisor_range>
  username: <mgmt_hypervisor2_username>
  password: <mgmt_hypervisor2_password>
  usage_tags:
    - MGMT
```

### 12.0.18 Deployment with Authentication

If you enable authentication, see the [authentication](#) page for information on how to configure Lightwave.

You can **add** the following directives to your deployment tree:

```
deployment:
  auth_enabled: true
  oauth_endpoint: 0.0.0.0
  oauth_port: 443
  oauth_tenant: <tenant>
  oauth_username: <user>
  oauth_password: <password>
  oauth_security_groups:
    - "oauth_tenant\<admin_group_name>"
```

### 12.0.19 Host Metadata

There are other optional metadata parameters that you can use for the management nodes:

Metadata	Description
ALLOWED_SERVICES	Whitelist of the permitted services
ALLOWED_DATASTORES	Whitelist of permitted datastores

Metadata	Description
ALLOWED_NETWORKS	Whitelist of allowed networks
MANAGEMENT_VM_CPU_COUNT_OVERWRITE	Number of CPUs for the management VMs
MANAGEMENT_VM_MEMORY_MB_OVERWRITE	Memory in MB; must be a power of 2
MANAGEMENT_VM_DISK_GB_OVERWRITE	Disk size

See below for an example of how to use these parameters. One difference to note here is that when you enable authentication, you have to specifically add the `ALLOWED_SERVICES` metadata value and set it to `Lightwave` for one of the `mgmt_hypervisors`. This is why you can see there are two sections of hosts in the yml file below: One section with only one `mgmt_hypervisor` on which lightwave will run, and another section with the rest of the `mgmt_hypervisors`, which would run the other Photon Controller services.

The `ALLOWED_SERVICES` parameter can accept the following values: `Lightwave`, `LoadBalancer`, `ManagementUi`, `PhotonControllerCore`.

hosts:

```
- metadata:
  ALLOWED_SERVICES: Lightwave
  ALLOWED_DATASTORES: "datastore1,datastore2"
  ALLOWED_NETWORKS: "Network1,Network2"
  MANAGEMENT_VM_CPU_COUNT_OVERWRITE:
  MANAGEMENT_VM_MEMORY_GB_OVERWRITE:
  MANAGEMENT_VM_DISK_GB_OVERWRITE:
  MANAGEMENT_DATASTORE: mgmt_datastore
  MANAGEMENT_PORTGROUP: mgmt_network
  MANAGEMENT_NETWORK_NETMASK: <mgmt_netmask>
  MANAGEMENT_NETWORK_DNS_SERVER: <mgmt_dns_server>
  MANAGEMENT_NETWORK_GATEWAY: <mgmt_gateway>
  MANAGEMENT_VM_IPS: <mgmt_vm1_ip>
  address_ranges: <one_of_the_mgmt_hypervisor_ips>
  username: <mgmt_hypervisor1_username>
  password: <mgmt_hypervisor1_password>
  usage_tags:
    - MGMT
- metadata:
  ALLOWED_DATASTORES: "datastore1,datastore2"
  ALLOWED_NETWORKS: "Network1,Network2"
  MANAGEMENT_VM_CPU_COUNT_OVERWRITE:
  MANAGEMENT_VM_MEMORY_GB_OVERWRITE:
  MANAGEMENT_VM_DISK_GB_OVERWRITE:
  MANAGEMENT_DATASTORE: mgmt_datastore
  MANAGEMENT_PORTGROUP: mgmt_network
  MANAGEMENT_NETWORK_NETMASK: <mgmt_netmask>
  MANAGEMENT_NETWORK_DNS_SERVER: <mgmt_dns_server>
  MANAGEMENT_NETWORK_GATEWAY: <mgmt_gateway>
  MANAGEMENT_VM_IPS: <mgmt_vm1_ip>
  address_ranges: <mgmt_hypervisor_range>
  username: <mgmt_hypervisor1_username>
  password: <mgmt_hypervisor1_password>
  usage_tags:
    - MGMT
```

Here are example values for the overwrite fields; you must use all three overwrite parameters for the values to be applied:

```
MANAGEMENT_VM_CPU_COUNT_OVERWRITE: 4
MANAGEMENT_VM_MEMORY_MB_OVERWRITE: 6144
MANAGEMENT_VM_DISK_GB_OVERWRITE: 80
```

## 13 Appendix III: Setting Up NSX Before Photon

Here's how to set up VMware NSX-T to work with Photon Controller. You must set up NSX-T before installing Photon Controller. After NSX-T is installed and configured to work with Photon Controller, you add the information about the NSX network to Photon Controller's YAML deployment file.

### 13.1 Prerequisites

- NSX-T 1.0.1. (In this document, *NSX* refers to NSX-T version 1.0.1.)
- Have dedicated ESXi hosts for NSX installation. The required version of ESXi is stated in the NSX Installation Guide. You cannot use the same ESXi hosts that you use for NSX for Photon Controller; after NSX is installed, you must install Photon Controller and Lightwave on separate ESXi hosts. Therefore, you will need at least 3 ESXi hosts: one for NSX, one for Photon Controller, and one for the Lightwave security system. NSX itself might require more than 1 ESXi host; see the NSX documentation.
- Read NSX release notes.
- Read the NSX Installation Guide to understand basic NSX concepts.
- Follow chapter 5 of the NSX Installation Guide to deploy NSX Manager onto an ESXi host.
- Follow chapter 6 of the NSX Installation Guide to deploy NSX Controller onto an ESXi host.
- Follow chapter 7 of the NSX Installation Guide to deploy NSX Edge onto an ESXi host.

### 13.2 References

- [NSX Release Notes](#)
- [NSX Installation Guide](#)
- [NSX Manager OVA](#)
- [NSX Controller OVA Download](#)
- [NSX Edge OVA](#)

### 13.3 Overview

Before diving into the details of NSX and DHCP configuration, you need to understand the relationship between NSX, DHCP, and Photon Controller.

#### 13.3.1 Fabric Topology

In Photon Controller 1.1, a single overlay transport zone is shared by all Photon Controller cloud hosts as well as a single Edge node. The overlay transport zone provides L2 and L3 connectivity between VMs through a logical switch and logical router.

A single VLAN transport zone is also created. The purpose of this VLAN transport zone is to provide connectivity between the logical network and the external, physical network. The Edge node joins both the overlay transport zone and VLAN transport zone.

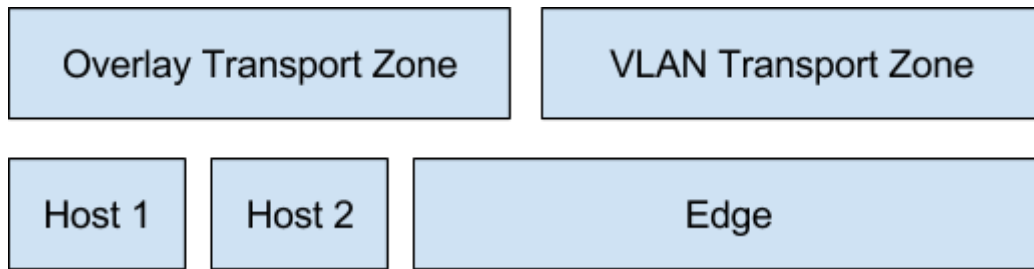


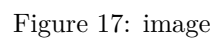
Figure 16: image

### 13.3.2 Virtual Network Topology

A single Tier-0 router is created to provide connectivity with the external network, as well as services like NAT, DHCP Relay, etc. For each virtual network, a Tier-1 router is created to provide L3 connectivity. If the network type is ROUTED, then the Tier-1 router is connected to the Tier-0 router. A logical switch is created for each virtual network as well to provide L2 connectivity. All VMs belonging to the same virtual network are attached to the corresponding logical switch.

A single DHCP server resides in a separate virtual network, which is manually set up. The DHCP server will handle DHCP requests from every virtual network, through the DHCP relay service configured on each Tier-1 router of the corresponding network. The DHCP server needs a private IP address as well as a public IP address.

Below is an example of setup with three virtual networks:





A single Tier-0 router is created and sits between the physical network and the virtual networks. For each virtual network created by Photon Controller, a single Tier-1 router and a single logical switch are created, as shown on the left side of the diagram. The Tier-1 router is configured with a private gateway IP address that is facing the VMs. A DHCP relay service is also attached to the gateway port, which points to the public IP address of the DHCP server.

The user sets up the dedicated DHCP network manually before deploying Photon Controller, which is shown on the right side of the diagram. The DHCP server has a static private IP address, such as 192.168.1.1. This IP address is accessible only inside the DHCP network, which is gated by the DHCP Tier-1 router. On the DHCP Tier-1 router, the gateway IP address is 192.168.1.253 on the gateway port. NAT rules are configured on the DHCP Tier-1 router to translate the public IP address assigned to the DHCP server to its private IP address.

## 13.4 NSX Configuration

After installing NSX components, you must configure NSX before deploying Photon Controller. The following steps are required:

- Create VLAN transport zone.
- Create overlay transport zone.
- Create uplink profile.
- Create IP address pool.
- Configure Edge transport node.
- Create Edge cluster.
- Create Tier-0 router.
- Connect Tier-0 router to the external network.

### 13.4.1 Create VLAN Transport Zone

Follow Chapter 9 of the NSX Installation Guide to create a VLAN transport zone. The name of the host switch can be arbitrary. The following screenshot provides an example:

New Transport Zone

×

Name: \*

tz-vlan

Description:

Host Switch Name: \*

vlan-host-switch

Traffic Type:

☐ Overlay

☒ VLAN

Save

Cancel

Figure 18: image

### 13.4.2 Create Overlay Transport Zone

Follow Chapter 9 of the NSX Installation Guide to create an overlay transport zone. The name of the host switch can be arbitrary. The following screenshot provides an example:

New Transport Zone

×

Name: \*

tz-overlay

Description:

Host Switch Name: \*

overlay-host-switch

Traffic Type:

☒ Overlay

☐ VLAN

Save

Cancel

Figure 19: image

### 13.4.3 Create Uplink Profile

The default uplink profile created by NSX defines a stand-by uplink, which is not supported by transport zone uplinks. Click Fabric -> Configuration -> Uplink Profiles to create a new uplink profile. For example:

New Uplink Profile

Name: \*

tz-uplink-profile

Description:

Teaming Policy: \*

Failover Order

LAGs

+ INSERT ROW

COLUMNS

Name \*

LACP Mode

LACP Load Balancing \*

Uplinks

LACP Time Out

Active Uplinks: \*

uplink-1

Standby Uplinks:

Transport VLAN:

0

MTU: \*

1600

Save

Cancel

Figure 20: image

102

#### **13.4.4 Create IP Address Pool**

The IP address pool provides internal IP addresses for NSX tunnel endpoints (i.e., ESXi hosts and Edge node). You will need to choose an internal IP address range that does not conflict with the IP addresses on the physical network.

Follow the NSX Installation Guide [Chapter 9] to create an IP address pool.

#### **13.4.5 Configure Edge Transport Node**

During NSX installation, an Edge entity automatically appears in the NSX Manager UI once the Edge is joined to the management plain. You can find the Edge entity under Fabric -> Nodes -> Edges.

To configure the Edge entity as a transport node, select the Edge entity, and then click Actions -> Configure as Transport Node.

Edit Transport Node - tn-edge-1

Name: \*

tn-edge

Transport Zones:

tz-overlay

tz-vlan

x

v

overlay-hostswitch

Edge Switch Name: \*

overlay-host-switch

Uplink Profile: \*

tz-uplink-profile

v

IP Pool:

tunnel-ip-pool

x

v

OR Create and Use a new IP Pool

Virtual NICs:

fp-eth0

v

uplink-1

v

vlan-uplink-hostswitch

x

Edge Switch Name: \*

vlan-host-switch

Uplink Profile: \*

tz-uplink-profile

v

IP Pool:

Select IP Pool

x

v

OR Create and Use a new IP Pool

Virtual NICs:

fp-eth1

v

uplink-1

v

Add New Node Switch

Save

Cancel

Figure 21: image



Note that you only need to choose an IP address pool for the overlay host switch. Also note that you need to choose the corresponding vnics for each vnic. Please refer to the NSX Installation Guide on how to choose and set up NSX Edge vnics.

#### **13.4.6 Create Edge Cluster**

To create an Edge cluster, click Fabric -> Configuration -> Edge Clusters.

Click the Edit button on the right to add the Edge transport node to the cluster.

#### **13.4.7 Create Tier-0 Router**

To create a Tier-0 router, click Routing -> Add -> Tier-0 Router. For example:

New Tier-0 Router

Tier-0 Router

Advanced

Name: \*

tier0-router

Description:

Edge Cluster: \*

edge-cluster

OR Create a New Edge Cluster

High Availability Mode:

☒ Active-Active

☐ Active-Standby

Save

Cancel

Figure 22: image

**Connect Tier-0 Router to External Network** To connect the Tier-0 router to the external network, you need to create a VLAN logical switch that connects the Tier-0 router. For example:

New Logical Switch

Name: \*

vlan-logical-switch

Description:

Transport Zone: \*

tz-vlan

Admin State: \*

Up

Switching Profiles Type: \*

None

Switching Profiles Id:

VLAN: \*

0

Save

Cancel

Figure 23: image

Create a port on the Tier-0 router that connects the VLAN logical switch:

New Router Port

Name: \*

to-vlan-logical-switch

Description:

Type:

☒ Uplink

☐ Downlink

Transport Node: \*

tn-edge-1

Logical Switch:

vlan-logical-switch

OR Create a New Switch

Logical Switch Port:

☒ Attach to new switch port

Switch Port Name:

to-tier0-router

☐ Attach to existing switch port

IP Address/mask: \*

10.178.56.240/23

Save

Cancel

Figure 24: image

Note that the IP address of the port needs to be an IP address that is accessible on the physical network.

Create a static route for the IP assigned to the port. Click the Tier-0 router, then click Routing -> Static Routes -> Add. Use 0.0.0.0/0 in the network field, and click Insert Row to add the gateway for the next hop.

## 13.5 DHCP Configuration

To configure the DHCP server, you will need to create a dedicated virtual network in NSX to host the server. This section describes how to set up such a network as well as how to configure the DHCP server.

- Note: The ESXi host that contains your DHCP server cannot be reused as a Photon Controller cloud host.

### 13.5.1 Create DHCP Server Virtual Network

As illustrated in the previous diagram, the DHCP server has its dedicated virtual network. The network is composed of a Tier-1 router and a logical switch.

To create a logical switch, click Switches -> Add. For example:

New Logical Switch

×

Name: \*

dhcp-server-switch

Description:

Transport Zone: \*

tz-overlay

▼

Admin State: \*

Up

Replication Mode:

☒ Hierarchical Two-Tier replication

☐ Head replication

Switching Profiles Type: \*

None

▼

Switching Profiles Id:

Save

Cancel

Figure 25: image



After creating the logical switch, create a Tier-1 router that connects the switch as well as the Tier-0 router:

New Tier-1 Router

Tier-1 Router

Advanced

Name: \*

dhcp-server-router

Description:

Tier-0 Router:

tier0-router

x

v

Edge Cluster:

edge-cluster

x

v

Members:

tn-edge

x

v

Preferred Member:

tn-edge

x

v

Save

Cancel

Figure 26: image

Create a logical router port that connects the Tier-1 router with the logical switch. Note that we chose 192.168.1.253/24 as the gateway IP address because the default private IP address of the DHCP server is 192.168.1.1. Change the IP address to match your environment.

New Router Port

Name: \*

to-dhcp-switch-port

Description:

Logical Switch:

dhcp-server-switch

✕

▼

OR Create a New Switch

Logical Switch Port:

☒ Attach to new switch port

Switch Port Name: to-dhcp-router-port

☐ Attach to existing switch port

IP Address/mask: \*

192.168.1.253/24

DHCP Service:

✕

▼

Save

Cancel

Figure 27: image

You will need to enable route advertisement so that other Tier-1 routers can reach the DHCP server. Click Routing -> Route Advertisement -> Edit, and enable all settings.

Edit Route Advertisement Configuration

X

Status:

Enabled

Advertise All NSX Connected Routes:

Yes

Advertise NAT Routes:

Yes

Advertise Static Routes:

Yes

Save

Cancel

Figure 28: image

Finally, create the NAT rules to map the public IP address to the private IP address. You will create a DNAT rule that maps your DHCP server's private IP address (Translated IP) to its public IP address (Destination IP). You will also create a SNAT rule that is the reserve of the DNAT rule (private IP address as Source IP and public IP address as Translated IP). Note that you will need to configure your physical router's routing table. The routing table needs to route traffic to the public IP address through the VLAN IP address you configured in the previous [Connect Tier-0 Router to External Network] section.

### 13.5.2 Deploy DHCP Server

Download the `dhcp-ova.ova` image file from the Photon Controller GitHub [downloads page](#) to deploy the DHCP server. Deploy the OVA file to an ESXi host. VMware recommends that you deploy the DHCP server on a separate host, though you can share the same host on which NSX Manager is installed. Note that during the deployment step for setting up the vnic of the DHCP VM, you should choose the NSX logical switch from the drop-down menu to connect the vnic.

Once deployed, before powering on the DHCP server, you need to add a CD-ROM to the VM and attach the `seed-dhcp-vm.iso` file to the CD-ROM. You can obtain the seed ISO on the Photon Controller [downloads page](#). If you did not add a CD-ROM and attach the seed ISO when you deployed the OVA, you can power off the DHCP server, add a CD-ROM drive to the VM, attach the seed ISO to it, and then power on the VM again.

The seed ISO automatically configures the VM's network settings and starts the daemon services. By default the private IP address of the DHCP server is set to 192.168.1.1. If you want a different private IP, you can log into the DHCP server VM, modify `/etc/systemd/network/10-static-<interface>.network`, and restart the network interface.

## 13.6 Deploy Photon Controller

You need to supply the NSX metadata to the Photon Controller deployment when virtual network is enabled. The following fields need to be specified under the `deployment` section of Photon Controller's YAML file used for deployment:

```
sdn_enabled: true
network_manager_address: NSX Manager IP address.
network_manager_username: NSX Manager admin username.
network_manager_password: NSX Manager admin password.
network_top_router_id: ID of Tier-0 router.
network_zone_id: ID of overlay transport zone.
network_edge_ip_pool_id: ID of the IP address pool.
network_host_uplink_pnic: Name of the physical NIC on the ESXi host that will be used as the overlay tu
network_ip_range: The global private IP address range (in CIDR format).
network_external_ip_range: The global floating IP address range (in start-end format).
network_dhcp_servers: List of DHCP server IP addresses. Currently a single entry of the DHCP server's p
```

Here is an example YAML deployment file that includes settings for NSX:

```
---
deployment:
  resume_system: true
  image_datastores: dc-test-rgb-1
  auth_enabled: false
  use_image_datastore_for_vms: true
  loadbalancer_enabled: true
  sdn_enabled: true
  network_manager_address: 203.0.113.155
```

```

network_manager_username: admin
network_manager_password: somesecret
network_top_router_id: 42e77601-f722-6f9b-yyy9-8st2b7b872rw
network_zone_id: rb18633z-5t92-9jj0-2k42-f3d44bv39g1g
network_edge_ip_pool_id: a077her2-054k-233f-8027-651v21f8q08w
network_host_uplink_pnic: vmnic1
network_ip_range: 192.168.3.0/24
network_external_ip_range: 198.51.100.129-198.51.100.234
network_dhcp_servers:
- 198.51.100.236
hosts:
- address_ranges: 203.0.113.55
  username: root
  password: yoursecret
  usage_tags:
  - MGMT
metadata:
  ALLOWED_DATASTORES: datastore1
  MANAGEMENT_DATASTORE: datastore1
  MANAGEMENT_PORTGROUP: Management VLAN
  MANAGEMENT_NETWORK_NETMASK: 255.255.254.0
  MANAGEMENT_NETWORK_DNS_SERVER: 203.0.113.1
  MANAGEMENT_NETWORK_GATEWAY: 203.0.113.253
  MANAGEMENT_VM_IPS: 203.0.113.120-203.0.113.122
- address_ranges: 203.0.113.57-203.0.113.58
  username: root
  password: yrsecret2
  usage_tags:
  - CLOUD

```

After NSX is installed, see the [NSX Admin Guide](#) for instructions on how to manage the virtual network.

## 14 Appendix IV: Release Notes

### 14.1 Photon Controller 1.1 Release Notes

#### 14.1.1 Supported Platforms

- VMware ESXi 6.0.0 Patch 201611001 (ESXi600-201611001), which you can find by searching for patches for ESXi 6.0.0 on the [My VMware Product Patches web site](https://my.vmware.com/group/vmware/patch) at <https://my.vmware.com/group/vmware/patch>.
- Lightwave version 1.0.1
- NSX-T version 1.0.1
- vSAN for Photon Platform version 1.1
- ESXi 6.5 is unsupported at this time.

#### 14.1.2 Security

- ESXi cloud hosts are joined to the Lightwave domain.
- Communication from Photon Controller to the Photon Controller agent on the ESXi hosts is secured with SSL.
- Requires Lightwave version 1.0.1.



- All SSL certificates for the Photon Platform management plane are issued by the Lightwave Certificate Authority.

### 14.1.3 vSAN Integration

- VMware vSAN for Photon Platform version 1.0.1 is required.
- Photon Controller has built-in support for vSAN for Photon Platform.
- Disk flavors can be tagged as vSAN-specific. Disks created with those flavors will be placed on a vSAN datastore.
- If a vSAN datastore is created after Photon Controller is installed, Photon Controller detects that vSAN datastore after 15 minutes.
- If you are planning to use vSAN, you must add a vSAN datastore to Photon Controller's deployment YAML file when you install Photon Controller.
- Known bug: vSAN performance service does not work properly.

### 14.1.4 NSX Integration

- VMware NSX-T version 1.0.1 is required.
- Photon Controller has built-in support for NSX.
- A custom DHCP server is provided to allow VMs to get IP addresses.
- Users can create multiple virtual networks, each associated with a single project. A single project allows multiple virtual networks.
- Users can create virtual machines that are attached to the virtual networks.
- Photon Controller orchestrates DHCP reservations.
- Users can assign floating IP addresses from a pool of addresses to a VM.
- Photon Controller can manage VLAN-backed networks or consume NSX networks, but not both.

### 14.1.5 Container Orchestration Frameworks

#### 14.1.5.1 Kubernetes

- Kubernetes has been upgraded to 1.4.3.
- Kubernetes UI and DNS are supported.
- Kubernetes can be configured to work with the Harbor Docker Registry.
- Many bug fixes and small improvements.
- Kubernetes currently works only with VLAN-backed networks.
- Known bug: If the Kubernetes master or etcd node fails, it will not be recreated. Kubernetes worker nodes are, however, recreated.

#### 14.1.5.2 Harbor Docker Registry

- The Harbor Docker Registry version 0.3.0 (docker register with integrated UI and role-based access) has been added as a new container cluster type. Although it is not a container orchestration cluster itself, it supports other clusters, especially Kubernetes.

#### 14.1.5.3 Other Cluster Types

- Support for Mesos and Docker Swarm remains experimental, and they can be deployed only through the API.

#### 14.1.6 UI

- Photon Controller has an entirely new UI.
- The UI supports differentiated access for system administrators, tenant administrators and project users.
- Users can create and manage Kubernetes clusters.
- Users can create and manage VMs.
- Known Issue: By default, the tokens granted by Lightwave last five minutes and are renewable. The UI and CLI do not renew the tokens automatically. As a workaround, you can extend the token lifetime through the Lightwave UI: Select “Policies & Configuration,” and then “Token Policy.” Edit “Max Token Lifetime” to provide a timeout in milliseconds.

#### 14.1.7 Photon Controller Agent

- The Photon Controller agent, which Photon Controller installs on the ESXi host, is deployed in conjunction with a Lightwave VIB. (VIB stands for vSphere Installation Bundle; it distributes an ESXi software package).
- The communication between Photon Controller and the agent is encrypted with SSL.
- The Photon Controller agent supports NSX.
- The Photon Controller agent supports vSAN.

#### 14.1.8 Deployment

- The installation process deploys Lightwave on a separate management VM. For instance, when you deploy four management VMs, one will have Lightwave and three will have Photon Controller.
- The installation process joins ESXi hosts to the Lightwave domain for secure communication.
- ESXi hosts should have static IP addresses and a non-default hostname.

#### 14.1.9 API

- When querying the API for the set of tenants or projects, a request now returns only the set of tenants or projects that the user is authorized to access.
- API support has been added for virtual networks.
- A new `/info` API provides information about whether SDN networking with NSX is enabled as well as version information about Photon Controller.

#### 14.1.10 Other

- A new [Quick Start Guide](#) has been added to our documentation. It’s available on our [GitHub wiki](#).