VMware Photon Platform 1.2 User Guide

Contents

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 Setting Up NSX.

2 Quick-Start Installation

2.1 Introduction

This guide explains how to install VMware Photon Controller for demonstration or trial purposes. As a quick start guide, it focuses on a minimal configuration to set up Photon Controller 1.2 with authentication in a controlled environment.

Setting up Photon Controller by following this guide prepares you to deploy a production-ready system that works in the context of your unique virtualized environment.

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

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 Version

This guide covers Photon Controller version 1.2. For earlier versions or a PDF, see Photon Platform Guides. To read the release notes, see Release Notes.

2.1.2 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 with SSH 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 coordinates workloads and allocates resources.
- A cloud host that resides on an ESXi host to run your users' VMs.

This guide installs Photon Controller on a single ESXi machine that holds the management plane, the cloud host, the load balancer, and the Lightwave security service. For a production system, you would deploy the management plane as a cluster of 3 VMs across multiple ESXi hosts. After you install the system, you can use the management plane to create tenants, quotas, and projects.

2.1.3 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.

The Photon Controller installation OVA installs Lightwave when it installs Photon Controller. Lightwave includes a DNS server that the Photon Controller management plane, cloud host, and other components use.

2.1.4 Assumptions

This guide assumes that an ESXi host is in place with the following attributes. If you do not have a computer running the ESXi operating system, you can obtain ESXi at the VMware vSphere Hypervisor Download Center. For help installing it, see the instructions at the download center and in the VMware vSphere 6 Documentation. The ESXi host can be either the licensed or the free version.

- It is running VMware ESXi 6.5 Patch 201701001 (ESXi650-201701001), which you can find by searching for patches for ESXi 6.5.0 on the My VMware Product Patches web site at https://my.vmware.com/group/vmware/patch. The patch's build number is 4887370.
- It contains no existing VMs or workloads 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 by vCenter.

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

If you plan to install Photon Controller with NSX-T, you must install NSX-T before installing Photon Controller; see Setting UP NSX.

2.1.5 Requirements

- Photon Controller version 1.2.
- At least three static IP addresses. One static IP address is required for the VM running the Photon Controller management node; a second static IP address is required for the VM running the Lightwave security service; and a third static IP address is required for the Photon load balancer.

After you install Photon Platform, you will need a workstation that can connect to a virtual machine on your ESXi host to run commands with the Photon Controller command-line interface (CLI). This guide uses a Linux workstation running Ubuntu 14.04 as an example. You can also install the Photon Controller CLI on a Microsoft Windows or Mac workstation.

2.2 Installing the Photon Controller Installation OVA

Download the installer for Photon Controller 1.2—`installer-ova-nv-1.2.0-dfea3bb.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 `CreateRegister 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 in the `Select storage` dialog, select a datastore; the examples in this guide assume that the datastore is named `datastore1`.

Click `Next` and in the `Deployment options` dialog, for `Network mappings`, use the default setting of `NAT VM Network`.

Move through the remaining dialog boxes by clicking `Next` to accept the default settings, and then click `Finish`. On the `Additional Settings` page, under `Options`, you can leave the fields empty.

When it finishes deploying, make sure it the power is on and the VM is running. Note its IP address. In a later step, you will connect to the VM by SSH to install the Photon Controller management VM, the Lightwave security service, and other components.

2.3 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 3 unused IP addresses that you can assign as static IP addresses. The static IP addresses are are for the VM

that will act as the Photon Controller management node, the VM that will run the Lightwave security service, and the VM that will act as the load balancer.
- Make sure that the pool of IP addresses available to `VM Network` is large enough to support provisioning several VMs later for working with Photon Platform.
- 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 with the ESXi command-line interface: 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)`.

2.4 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 uses the values in the YAML file to identify the correct networking settings, locate the datastore, and set up the management node and cloud host as well as the Lightwave security service.

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.

2.4.1 YAML Rules

Indentation–defined as zero or more space characters at the start of a line–determines the structure of a block of YAML code. You must indent each line further than its parent, and all sibling nodes must adhere to the exact same level of indentation.

Do not use tabs or tab characters to indent lines in your YAML file. Tabs and improper indentation are the most likely causes of superficial YAML errors when the installer processes a YAML configuration file.

For example, if you copy one of the example YAML code blocks in this document and paste it into a new file, it might throw errors when the installer processes it. Make sure that the indentation is correct and that the file contains no tabs. See the YAML specification.

You can validate your YAML configuration file at YAML Lint: http://www.yamllint.com/.

2.4.2 The Anatomy of a Configuration File

The template contains four main sections:

1. The `compute` section lists networking information for all ESXi hosts that are to be included in the deployment of Photon Platform.
2. The `lightwave` section specifies the target ESXi host and the configuration for the Lightwave security service. It also includes the credentials for initially logging on to Photon Platform as an administrator.
3. The `photon` section specifies the ESXi machines that will host the image stores, cloud hosts, and management VMs for Photon Controller.
4. The `loadBalancer` section sets the configuration for the loadbalancer that Photon Controller uses for incoming traffic.

Here is a minimal template with descriptions of each key-value pair. The values of key-value pairs are set in double quotation marks.

```
compute:
  hypervisors:
    esxi-1:
      hostname: The host name of the target ESXi hypervisor that you want to include
                in the Photon Controller cluster.
      ipaddress: The IP address of the target ESXi hypervisor.
      dns: The static IP address that you are assigning to the Lightwave VM in the
           lightwave section of the YAML configuration.
      credential:
        username: The name of the root account on the target ESXi hypervisor;
                  for example, "root"
        password: The password for the root account of the target ESXi hypervisor.
lightwave:
  domain: The domain name that you want to set for the Lightwave domain;
          example, "example.com"
  credential:
    username: The default user name that you want set as the administrator of the
              Lightwave service. It must be all lowercase letters;
              example, "administrator"
    password: The default password that you want to set for the user name.
  controllers:
    lightwave-1:
      site: Insignificant designation of your site's location;
            this value is not acted on by the installer; example: "nydc"
      appliance:
        hostref: The name of the ESXi hypervisor on which you want Lightwave
                 to be installed.
        datastore: "datastore1"
        credential:
          username: The name of the root account on the target ESXi hypervisor.
          password: The password for the root account.
        network-config:
          network: "NAT=VM Network"
          type: "static"
          hostname: The fully qualified domain name that you want to set as
                    the hostname of the VM on which Lightwave resides.
          ipaddress: The static IP address that you want to assign
                     to the VM running Lightwave.
          dns: The IP address of your internal corporate DNS server.
          ntp: The IP address of the NTP server that the Lightwave VM will use.
          netmask: The netmask that the Lightwave VM will use.
          gateway: The gateway IP address that you want the Lightwave VM to use.
photon:
  imagestore:
    img-store-1:
      hostref: The target ESXi hypervisor on which you want Photon Controller image
               datastore to reside.
               The hypervisor must be included in the compute section's
               list of hypervisors.
      datastore: The datastore on an ESXi hypervisor for images for VMs and
                 Kubernetes clusters, etc. The datastore must be unique; that is,
                 the name of each datastore must be unique among the ESXi hosts
                 in the deployment. If there are, for instance, two ESXi hosts
                 in the deployment, and the first ESXi host's datastore is named
```

```
                datastore1, then the second ESXi host's datastore must have a
                different name, such as datastore2.
      enableimagestoreforvms: Allows the datastore that is set as the image
                              datastore to be used by VMs. This value must be
                              set to true if there is only one ESXi host in the
                              deployment.
  cloud:
    hostref1: "esxi-17"
  syslog:
    ipaddress: The IP address of the syslog server to which you want to send Photon
               Controller log files. The syslog entry and its ipaddress key-value pair
               are optional. You can remove both lines.
  controllers:
    pc-1:
      appliance:
        hostref: The target ESXi hypervisor on which you want the Photon Controller
                 management VM to reside. The hypervisor must be included in the
                 compute section's list of hypervisors.
        datastore: The datastore that this VM will use; it must be the name of the
                   datastore on the ESXi host referenced by the value of the
                   hostref in the previous line.
        credential:
          username: The name of the root account of the target hypervisor.
          password: The password for the root account.
        network-config:
          network: "NAT=VM Network"
          type: "static"
          hostname: The fully qualified domain name that you want to set as
                    the hostname of the Photon Controller management VM.
          ipaddress: The static IP address you want to assign to the management VM.
          netmask: The netmask that the  management VM will use.
          dns: The IP address of the DNS server that the management VM will use.
          ntp: The IP address of the NTP server that the management VM will use.
          gateway: The gateway IP address that you want the management VM to use.
loadBalancer:
  load-balancer-1:
    appliance:
      hostref: The name of the ESXi hypervisor on which you want the load
               balancer to reside. The hypervisor must be included in the
               compute section's list of hypervisors.
      datastore: "datastore1"
      credential:
        username: "root"
        password: "secret$11"
      network-config:
        network: "NAT=VM Network"
        type: "static"
        hostname: The fully qualified domain name that you want to assign as the
                  host name of the load balancer; example: "lb-1.eg.example.com"
        ipaddress: The static IP address that you want to assign to the load balancer.
        netmask: The netmask that the load balancer will use.
        dns: The IP address of the DNS server that the load balancer will use.
        ntp: The IP address of the NTP server that the load balancer will use.
        gateway: The gateway IP address that you want the load balancer to use.
```

2.4.3 DNS

The value that you must set for the `dns` entry for each section except `lightwave` is counterintuitive: Because Photon Controller uses the DNS server that is included with Lightwave, the DNS entry for each section except `lightwave` must be the static IP address that you are assigning to the Lightwave VM in the `lightwave` section of the YAML configuration.

Here is an example that shows the value of the Lightwave VM's `dns` key and the VM's `ipaddress` key. For the `lightwave` section only, the `dns` key is to be set to the IP address of your internal corporate network's DNS server. The value of the `dns` key for all other sections is to be set to the same value as IP address of the Lightwave VM:

```
lightwave-1:
...
    network-config:
      network: "NAT=VM Network"
      type: "static"
      hostname: "lightwave-1.example.com"
      ipaddress: "198.51.100.212"
      dns: "192.0.2.1"
```

In the example configuration file that follows, note that the `dns` key *for each section except Lightwave* is set to the same static IP address—the value of the `ipaddress` key in the `lightwave` section.

2.4.4 Example Basic Configuration File

Here is an example configuration file with the minimum information necessary for a basic deployment on a single ESXi hypervisor.

There is also a full example YAML configuration file on the installer VM. The password for the installer VM is `changeme`:

```
ssh root@198.51.100.212
cd /opt/vmware/photon/controller/share/config/
ls
log4j.properties  pc-config.yaml
```

```
compute:
  hypervisors:
    esxi-1:
      hostname: "esxi-1"
      ipaddress: "198.51.100.44"
      dns: "198.51.100.212"
      credential:
        username: "root"
        password: "secret$1"
lightwave:
  domain: "example.com"
  credential:
    username: "administrator"
    password: "Secret1!"
  controllers:
    lightwave-1:
      site: "wdc"
      appliance:
        hostref: "esxi-1"
```

```
        datastore: "datastore1"
        credential:
          username: "root"
          password: "secret$11"
        network-config:
          network: "NAT=VM Network"
          type: "static"
          hostname: "lightwave-1.example.com"
          ipaddress: "198.51.100.212"
          dns: "192.0.2.1"
          ntp: "198.51.100.1"
          netmask: "255.255.0.0"
          gateway: "198.51.100.253"
photon:
  imagestore:
    img-store-1:
      datastore: "datastore1"
      enableimagestoreforvms: "true"
  cloud:
    hostref1: "esxi-1"
  controllers:
    pc-1:
      appliance:
        hostref: "esxi-1"
        datastore: "datastore1"
        credential:
          username: "root"
          password: "secret$11"
        network-config:
          network: "NAT=VM Network"
          type: "static"
          hostname: "pc-1.example.com"
          ipaddress: "198.51.100.208"
          netmask: "255.255.0.0"
          dns: "198.51.100.212"
          ntp: "198.51.100.1"
          gateway: "198.51.100.253"
loadBalancer:
  load-balancer-1:
    appliance:
      hostref: "esxi-1"
      datastore: "datastore1"
      credential:
        username: "root"
        password: "secret$11"
      network-config:
        network: "NAT=VM Network"
        type: "static"
        hostname: "lb-1.example.com"
        ipaddress: "198.51.100.207"
        netmask: "255.255.0.0"
        dns: "198.51.100.212"
        ntp: "198.51.100.1"
        gateway: "198.51.100.253"
```

Note: If NSX-T is installed and you want to include the NSX-T network information in the YAML configuration file, see Integrating with NSX.

For your first test deployment, you can either use the example YAML configuration file on the installer VM or copy the template above, paste it into a file, remove the tabs, and save the file as `config1.yaml`.

The full example YAML configuration file resides on the installer VM as `pc-config.yaml`. The password for the installer VM is `changeme`:

```
ssh root@198.51.100.212
cd /opt/vmware/photon/controller/share/config/
ls
log4j.properties  pc-config.yaml
```

You can obtain most of the information to fill out the template by connecting to your ESXi host with the vSphere Web Client. Replace the IP addresses and the user names and password with those from your ESXi host and its network. The IP addresses for the Photon management VM and the Lightwave VM should be static IP addresses that are available for use by the ESXi host. You might have to obtain static IP addresses from your network administrator.

2.5 Deploying Photon Platform

After you modify the YAML configuration template to match your environment, transfer the file to the Photon installer VM by using scp or sftp. The password for the installer VM is `changeme`. Example:

```
scp config1.yaml root@198.51.100.212:/tmp/config1.yaml
Password: changeme
config1.yaml                                  100% 1839     1.8KB/s   00:00
```

Next, connect to the installer VM by using SSH; example:

```
ssh root@198.51.100.212
```

The installer VM includes a command-line utility that sets up Photon Platform. Take a moment to view its help by running the following command:

```
/opt/vmware/photon/controller/bin/photon-setup
```

Here's what the output looks like:

```
Usage: photon-setup <component> <command> {arguments}
Component:
    platform:      Photon Platform including multiple components
    controller:    Photon Controller
    lightwave:     Lightwave
    controller:    Photon Controller Cluster
    agent:         Photon Controller Agent
    vsan:          Photon VSAN Manager
    dhcp:          DHCP Instance
    load-balancer: Load balancer
    help:          Help
Command:
    install:   Install components
    help:      Help about component
Run 'photon-setup <component> help' to find commands per component
```

To install the platform from your YAML configuration file, you will use the `photon-setup platform` command. Here's what its help output looks like:

```
/opt/vmware/photon/controller/bin/photon-setup platform help
Usage: photon-setup platform <command> <arguments>
Command:
    install
    help
platform install:
    -config <path to config file>
```

Now run the following command to install the Photon Platform components specified in the YAML file:

```
/opt/vmware/photon/controller/bin/photon-setup platform install -config /tmp/config1.yaml
```

The installation takes a few minutes. If the deployment is unsuccessful, see the section on troubleshooting.

If the deployment is successful, leave the Photon Platform installer VM in place; do not delete it. You are might need it later to add additional ESXi hosts to the cluster.

2.6 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 the Photon Controller management VM that you installed on an ESXi host. Make sure to download the version of the CLI tool that coincides with the version of Photon Controller you are installing.

```
https://github.com/vmware/photon-controller/releases
```

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
```

You can also install the Photon Controller CLI on a Microsoft Windows or Mac workstation.

2.7 Connecting to the Load Balancer

To manage Photon Controller, you connect to the load balancer from your workstation by using either the web interface or the command-line interface.

2.7.1 Using the CLI

After the installer deploys Photon Controller, you can connect to the load balancer to create tenants, quotas, and projects. But first, you will need the load balancer's IP address from your YAML file.

You connect to Port 443 of the IP address of the load balancer by running the following `photon target set` command with the `-c` option:

```
photon target set -c https://<ip-of-load-balancer>:443
```

After you set the target, you can log in by using the credentials that you set in the `lightwave` section of the YAML configuration file; in the example, it looked like this:

```
lightwave:
  domain: "example.com"
  credential:
    username: "administrator"
    password: "Secret1!"
```

Here's the syntax of the command to log in:

```
photon target login --username <username>@<lightwave-domain> --password 'Your$ecret1!'
```

Here is an example. In the sample YAML file, the `username` of the Lightwave `credential` is set to `administrator` and the domain is set to `example.com`. The `photon target login` uses the Lightwave domain and credentials to authenticate the user with the Lightwave security service.

```
photon target login --username administrator@example.com --password 'Secret1!'
```

Now that you are logged in, you can check the status of Photon Controller:

```
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 get more information about the deployment by running the following command:

```
photon deployment show
Deployment ID: default
  State:                          READY
  Image Datastores:               [datastore1]
  Use image datastore for vms: true
  Syslog Endpoint:                -
  Ntp Endpoint:                   -
  LoadBalancer:
    Enabled:                      false
  Auth:
    Endpoint:                     198.51.100.212
    Tenant:                       example.com
    Port:                         443
    Securitygroups:               [example.comAdministrators]
  Stats:
    Enabled:               false
  Migration status:
    Completed data migration cycles:          0
    Current data migration cycles progress:   0 / 0
    VIB upload progress:                      0 / 0
  Cluster Configurations:
    No cluster is supported
  Job  VM IP(s)  Ports
  VM IP  Host IP  VM ID  VM Name
```

To start working with Photon Platform, see the Command-Line Cheat Sheet and the other documentation on the Photon Controller GitHub Wiki. To create a tenant, for instance, see Working with Tenants.

2.7.2 Using the Web Interface

You can log in to the web interface to view a range of information and to create tenants, allocate resources, establish projects, and spin up clusters. To log in the web interface, connect to the load balancer over HTTPS by using its IP address plus Port 4343:

```
https://<ip-address-of-load-balancer>:4343
```

The load balancer redirects your browser to the IP address of the Lightwave security service. Log in by using the credentials that you set in the `lightwave` section of the YAML configuration file; for example:

```
adminsitrator@example.com
Secret1!
```

2.8 Creating a Kubernetes Cluster

To see the power of Photon Platform, you can create a Kubernetes cluster. For instructions, see Creating a Kubernetes Cluster on the Photon Controller GitHub wiki.

2.9 Troubleshooting

You can troubleshoot by looking at the installer's logs. The most likely cause of a failure is that a static IP address that you tried to assign to a VM in the Photon Controller management plane is in use, unavailable, or unreachable. Another possible cause is that the DNS entries are incorrect in the YAML configuration file.

2.9.1 Log Files for the Installer

The log files for the deployment reside in the following location on the installer VM. The password for the root account is `changeme`.

- /var/log/photon-installer.log

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

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

The log files on the Photon Controller management VMs might contain useful troubleshooting information about API calls that fail, such as the call to add an ESXi host to the cluster. The management VM's log resides at this location:

- /var/log/photon-platform/photon-controller.log

You can access the log file through the VM's console in the vSphere Web Client. To connect to the management VM with SSH to view the log, you must first connect to the console and change the SSH configuration to permit root login; see Permitting Root Login with SSH.

2.9.2 General Troubleshooting Steps for Unknown Problems

If there is no immediate indication of the issue that's blocking your installation, here are some things to check on the ESXi hosts to try to identify the problem.

First, make sure the Photon Controller agent is running:

```
/etc/init.d/photon-controller-agent status
```

Second, check whether each ESXi host has joined the Lightwave domain:

```
/usr/lib/vmware/vmafd/bin/domainjoin
```

Third, determine whether the SSL from the Lightwave certificate authority has been successfully deployed. The SSL certificates reside in `/etc/vmware/ssl`. Run the following command and check the Subject and X509v3 Subject Alternative Name fields. The rui.crt should have been replaced by the certificate from the Lightwave certificate authority:

```
openssl x509 -in /etc/vmware/ssl/rui.crt -noout -text
```

Examine the output of the command to make sure the Issuer field shows that the Photon Controller agent is using the certificate from the Lightwave certificate authority. Is the Subject Name and Subject Alt Name on the SSL certificate corresponding to the ESXi's host name and its IP address?

If everything is OK, return to Photon Controller. Check the logs to see if there is a connection error to the Photon agent? Is there an issue registering the agent in the Photon cloud store?

2.9.3 Installing in Memory-Constrained Environments

If you are trying to install Photon Platform on ESXi hosts with memory constraints, you can specify the amount of RAM and CPUs that the VM will use on the target ESXi host by using the `memoryMB` key and the `cpus` key. Setting the values for these keys can help overcome out-of-memory or not-enough-memory errors.

However, the values must be, at a minimum, 2048 megabytes of memory and 2 CPU cores. Here is an example that uses the minimum settings for memory and CPUs for a Lightwave VM:

```
controllers:
  lightwave-1:
    site: "New York"
    appliance:
      hostref: "esxi-1"
      datastore: "datastore1"
      memoryMb: 2048
      cpus: 2
      credential:
        username: "root"
        password: "Your$ecret1"
      network-config:
        type: "static"
        hostname: "lightwave-1.example.com"
        ipaddress: "198.51.100.12"
        network: "NAT=VM Network"
        dns: "198.51.100.12,198.51.100.13"
        ntp: "192.0.2.1"
        netmask: "255.255.252.0"
        gateway: "198.51.100.253"
```

2.9.4 Retrying the Installation

If Photon Controller fails to install successfully, modify the YAML configuration file and try again by rerunning the installation command with the corrected YAML file.

2.9.5 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.

2.10 Deploying a Production-Level Platform

Setting up Photon Controller by following this quick start guide prepares you to deploy a production-ready system that works within the context of your unique virtualized environment. The key is to take the knowledge that you gained from a minimal installation and to apply it to modifying the YAML configuration file for a production-level deployment.

A production-level deployment should contain at least three VMs dedicated to the Photon Controller management plane—and all the management VMs should be assigned static IP addresses. The management VMs should also reside on different ESXi hypervisors to distribute the load and to ensure high availability.

In addition, a production-level deployment can contain more ESXi hypervisors to add compute and storage resources to the cluster.

A production-level deployment of Photon Platform should include VMware NSX-T and VMware vSAN. NSX-T provides software-defined virtualized networking, and vSAN furnishes a virtual storage area network. Both are addressed in later sections.

Here is a YAML configuration file that includes a number of ESXi hosts and three Photon Controller management plane VMs. Notice how the three management VMs—`pc1`, `pc2`, and `pc3`—are each installed on a different ESXi hypervisor.

```
compute:
  hypervisors:
    esxi-24:
      hostname: "esxi-24"
      ipaddress: "198.51.100.48"
      dns: "198.51.33.10"
      credential:
        username: "root"
        password: "secret$1"
    esxi-17:
      hostname: "esxi-17"
      ipaddress: "198.51.100.41"
      dns: "198.51.33.10"
      credential:
        username: "root"
        password: "secret$1"
    esxi-18:
      hostname: "esxi-18"
      ipaddress: "198.51.100.42"
      dns: "198.51.33.10"
      credential:
        username: "root"
        password: "secret$1"
    esxi-19:
      hostname: "esxi-19"
      ipaddress: "198.51.100.43"
      dns: "198.51.33.10"
      credential:
        username: "root"
        password: "secret$1"
    esxi-20:
      hostname: "esxi-20"
      ipaddress: "198.51.100.44"
      dns: "198.51.33.10"
```

```
      credential:
        username: "root"
        password: "secret$1"
    esxi-21:
      hostname: "esxi-21"
      ipaddress: "198.51.100.45"
      dns: "198.51.33.10"
      credential:
        username: "root"
        password: "secret$1"
    esxi-22:
      hostname: "esxi-22"
      ipaddress: "198.51.100.46"
      dns: "198.51.33.10"
      credential:
        username: "root"
        password: "secret$1"
lightwave:
  domain: "example.com"
  credential:
    username: "administrator"
    password: "VMware123$"
  controllers:
    lightwave-1:
      site: "wdc"
      appliance:
        hostref: "esxi-24"
        datastore: "datastore1"
        credential:
          username: "root"
          password: "secret$11"
        network-config:
          network: "NAT=VM Network"
          type: "static"
          hostname: "lightwave-1.example.com"
          ipaddress: "198.51.33.10"
          dns: "192.0.2.1"
          ntp: "198.51.100.1"
          netmask: "255.255.240.0"
          gateway: "198.51.100.253"
photon:
  imagestore:
    img-store-1:
      hostref: "esxi-17"
      datastore: "cloud1-ds-17_20,cloud1-ds-21_24"
  cloud:
    hostref1: "esxi-17"
    hostref2: "esxi-18"
    hostref3: "esxi-19"
    hostref4: "esxi-20"
    hostref5: "esxi-21"
    hostref6: "esxi-22"
  controllers:
    pc-1:
```

```
      appliance:
        hostref: "esxi-17"
        datastore: "cloud1-ds-17_20"
        credential:
          username: "root"
          password: "secret$11"
        network-config:
          network: "NAT=VM Network"
          type: "static"
          hostname: "pc-1.example.com"
          ipaddress: "198.51.33.20"
          netmask: "255.255.240.0"
          dns: "198.51.33.10"
          ntp: "198.51.100.1"
          gateway: "198.51.100.253"
    pc-2:
      appliance:
        hostref: "esxi-18"
        datastore: "cloud1-ds-17_20"
        credential:
          username: "root"
          password: "secret$11"
        network-config:
          network: "NAT=VM Network"
          type: "static"
          hostname: "pc-2.example.com"
          ipaddress: "198.51.33.21"
          netmask: "255.255.240.0"
          dns: "198.51.33.10"
          ntp: "198.51.100.1"
          gateway: "198.51.100.253"
    pc-3:
      appliance:
        hostref: "esxi-21"
        datastore: "cloud1-ds-21_24"
        credential:
          username: "root"
          password: "secret$11"
        network-config:
          network: "NAT=VM Network"
          type: "static"
          hostname: "pc-3.example.com"
          ipaddress: "198.51.33.22"
          netmask: "255.255.240.0"
          dns: "198.51.33.10"
          ntp: "198.51.100.1"
          gateway: "198.51.100.253"
loadBalancer:
  load-balancer-1:
    appliance:
      hostref: "esxi-17"
      datastore: "datastore1"
      credential:
        username: "root"
```

```
    password: "secret$11"
  network-config:
    network: "NAT=VM Network"
    type: "static"
    hostname: "lb-1.example.com"
    ipaddress: "198.51.33.19"
    netmask: "255.255.240.0"
    dns: "198.51.33.10"
    ntp: "198.51.100.1"
    gateway: "198.51.100.253"
```

2.10.1 Integrating with NSX

Here's an example of the YAML code block for NSX. The value for the `ipaddress` key in the `nsxconfig` section is the IP address of your NSX Network Manager.

```
nsxconfig:
  ipaddress: "203.0.113.220"
  credential:
    username: "admin"
    password: "secret$1"
  privateIpRootCidr: "192.168.1.0/24"
  floatingIpRootRange: "203.0.113.160-203.0.113.170"
  t0RouterId: "d3b1a8gr-6c58-2062-2562-2drc8977e414"
  edgeClusterId: "55338b48-4r72-38b0-8d4r-65b29084c99a"
  overlayTransportZoneId: "dg821bea-c5r3-34b2-a32g-b02d44726d24"
  tunnelIpPoolId: "b4h8c34d-7714-507c-78g2-ef93b6b2db2a"
  hostUplinkPnic: "vmnic4"
  hostUplinkVlanId: "0"
  dnsServerAddresses: "198.51.100.12"
```

2.10.2 Integrating with vSAN

Adding VMware vSAN to Photon Platform 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 you install Photon Platform, you can add a virtual storage area network by deploying vSAN. For installation instructions, see Deploying vSAN for Photon Platform.

2.11 Creating Accounts in Lightwave

After you deploy Photon Platform, you can log in to the Lightwave service with SSH and create additional users and groups.

Before you can log in with SSH, however, you must access the VM through its console in ESXi and modify the VM's SSH configuration file to permit root login.

To permit root login over SSH, open `/etc/ssh/sshd_config` with the vim text editor and set all three instances of `PermitRootLogin` to `yes`. After you modify and save the SSH daemon's configuration file, you must restart the sshd daemon for the changes to take effect:

```
systemctl restart sshd
```

Then connect to the Lightwave VM with SSH using the root account and password that you set in the YAML file for the Lightwave VM; example:

```
ssh root@<IPaddressOfLightwaveVM>
```

After you log in to the Lightwave VM with SSH, change directories:

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

Using the Lightwave administrator password that you defined in the YAML configuration file when prompted, run the following Lightwave directory commands to create a group, 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
```

When you are done, type `exit` again to leave the SSH session.

2.11.1 Connecting to the Lightwave Web Interface

To connect to the Lightwave web interface, you must first add a Lightwave entry to the `/etc/hosts` file of your Linux workstation so that your browser can use the hostname that you set for the Lightwave VM instead of its IP address.

Here's an example of how to add a Lightwave entry to `/etc/hosts`:

```
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting.  Do not change this entry.
##
127.0.0.1   localhost
255.255.255.255 broadcasthost
::1             localhost
198.51.33.10    lightwave-1.example.com
```

You can now connect to the Lightwave VM by connecting to its full hostname over HTTPS; example:

```
https://lightwave-1.example.com
```

2.11.2 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.

2.11.3 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 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

- Setting Security Groups for Administrators, Tenants, and Project Users
- Connecting to the Load Balancer for Secure Login
- Tenants, Quotas, 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 quotas, 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. Connect over SSH to the virtual machine running Lightwave and use the directory command-line utility to create users and groups.

2. Create a system administrator group and user for Photon Controller so that you can log on to Photon Controller and be authenticated with Lightwave.
3. Optionally create additional users and groups in Lightwave by using the Lightwave directory utility.
4. 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

You can log in to the Lightwave service with SSH and create additional users and groups.

Before you can log in with SSH, however, you must access the VM through its console in ESXi and modify the VM's SSH configuration file to permit root login.

To permit root login over SSH, open `/etc/ssh/sshd_config` with the vim text editor and set all three instances of `PermitRootLogin` to `yes`. After you modify and save the SSH daemon's configuration file, you must restart the sshd daemon for the changes to take effect:

```
systemctl restart sshd
```

Then connect to the Lightwave VM with SSH using the root account and password that you set in the YAML file for the Lightwave VM; example:

```
ssh root@<IPaddressOfLightwaveVM>
```

After you log in to the Lightwave VM with SSH, change directories:

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

Using the Lightwave administrator password that you defined in the YAML configuration file when prompted, run the following Lightwave directory commands to create a group, 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
```

When you are done, type `exit` again to leave the SSH session.

Connecting to the Lightwave Web Interface

You can also create users and groups by the Lightwave web interface. To connect to it, you must first add a Lightwave entry to the `/etc/hosts` file of your Linux workstation so that your browser can use the hostname that you set for the Lightwave VM instead of its IP address.

Here's an example of how to add a Lightwave entry to `/etc/hosts`:

```
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting.  Do not change this entry.
##
127.0.0.1   localhost
255.255.255.255 broadcasthost
::1             localhost
198.51.33.10    lightwave-1.example.com
```

You can now connect to the Lightwave VM by connecting to its full hostname over HTTPS; example:

```
https://lightwave-1.example.com
```

3.1.5 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 administrator password defined in the your YAML deployment template (`L1ghtwave!` 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
```

3.1.6 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
```

3.1.7 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.

3.1.8 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.

3.1.9 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
```

3.1.10 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, quotas, 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 that you created in the Lightwave directory. Combine the account name with the domain name defined in your YAML template. Lightwave authenticates the user.

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

Here is an example:

```
photon target login --username pc-admin@example.com --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.

3.1.11 Tenants, Quotas, 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, quotas, 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 quotas. A quota 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 quota
vm.memory	integer	GB	The amount of virtual memory, in gigabytes, that can be consumed by all virtual machines in all projects using this quota
vm.cpu	integer	COUNT	The number of virtual CPU cores that can be consumed by all virtual machines in all projects using this quota

A quota is scoped to a singe tenant, persists for the life of the tenant, and is removed when the tenant is deleted.

3.1.12 Projects

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

Tenant administrators create projects based on a quota and can impose project-specific limits on the resources. A project can be granted the right to use all or part of a quota'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, quotas, 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
/datastores	List datastores in use by Photon Controller
/deployments	List Photon Controller deployments
/hosts	List hosts under control of Photon Controller
/portgroups	List portgroups available
/status	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
/flavors	Manage flavors	List/show flavors
/networks	Manage networks	List/show networks
/tenants	Manage tenants	List/show tenants
/tenants/quotas	Manage quotas	List/show quotas

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 quotas
- 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
/tenants/MY-TENANT/set_security_groups	Manage security groups for MY-TENANT
/tenants/MY-TENANT/projects	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
/projects/MY-PROJECT/clusters	Manage container clusters for MY-PROJECT
/projects/MY-PROJECT/disks	Manage disks for MY-PROJECT
/projects/MY-PROJECT/vms	Manage VMs for MY-PROJECT
/projects/MY-PROJECT/set_security_groups	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
```

Connect to the IP address of the load balancer by appending Port `443`:

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

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@example --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, quotas, and projects.

4.3 Connecting to the Web UI

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

`https://<ip-address-of-load-balancer>:4343`

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

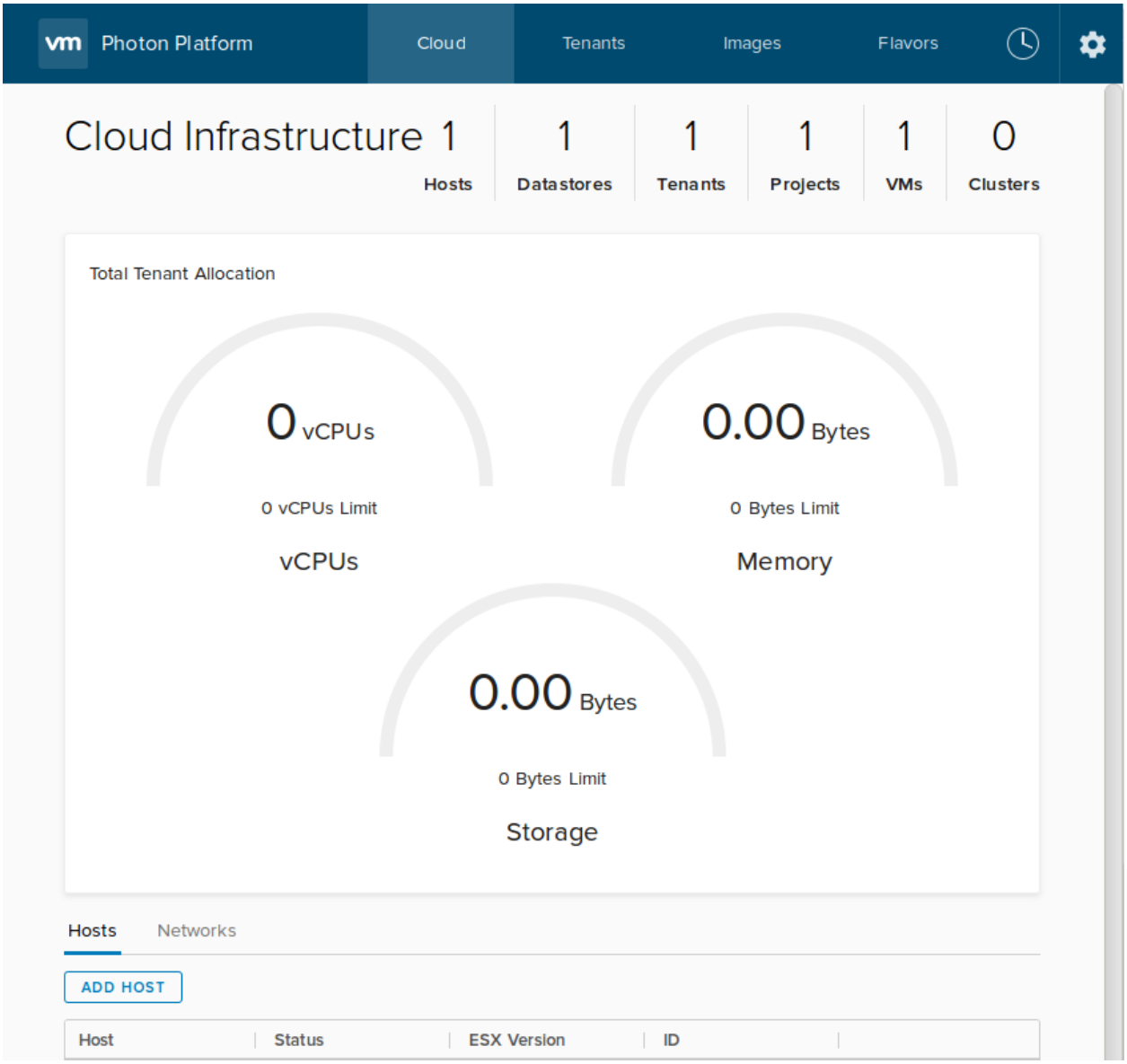

Figure 1: Photon Controller Web UI

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 tenant quota --help
photon service create -h
```

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

```
photon -h
NAME:
   photon - Command line interface for Photon Controller
USAGE:
   photon [global options] command [command options] [arguments...]
VERSION:
   Git commit hash: 11f5e4c
COMMANDS:
     auth              options for auth
     system            options for system operations
     target            options for target
     tenant            options for tenant
     host              options for host
     datastore         options for datastore
     deployment        options for deployment
     image             options for image
     task              options for task
     flavor            options for flavor
     project           options for project
     disk              Options for persistent disks
     vm                options for vm
     service, cluster  Options for services
     router            options for router
     subnet            options for subnet
     zone              options for 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 value, -l value  write logging information into a logfile at
                               the specified path
   --output value, -o value    select output format
   --detail, -d                print the current target, user, tenant, project
   --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

To manage Photon Controller, you connect to the load balancer from your workstation by using the command-line interface. You can then create tenants, quotas, projects, and Kubernetes clusters. (You can obtain the load balancer's IP address from your YAML deployment file.)

You connect to Port 443 of the IP address of the load balancer by running the following `photon target set` command with the `-c` option:

```
photon target set -c https://<ip-of-load-balancer>:443
```

After you set the target, you can log in by using the credentials that you set in the `lightwave` section of the YAML configuration file. Here's an example:

```
lightwave:
  domain: "example.com"
  credential:
    username: "administrator"
    password: "Secret1!"
```

Here's the syntax of the command to log in:

```
photon target login --username <username>@<lightwave-domain> --password 'Your$ecret1!'
```

Here is an example. In the sample YAML code block above, the `username` of the Lightwave `credential` is set to `administrator` and the domain is set to `example.com`. The `photon target login` uses the Lightwave domain and credentials to authenticate the user with the Lightwave security service.

```
photon target login --username administrator@example.com --password 'Secret1!'
```

Once logged in, you can check the status of Photon Controller:

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

You can get more information about the deployment by running the following command:

```
photon deployment show
```

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 quotas:

```
photon tenant create <name_of_tenant>
photon tenant list
photon tenant show <tenant-ID>
photon tenant quota show <tenant-id>
photon project quota show <project-id>
photon tenant quota set ...
photon project quota set ...
```

Here are some examples:

```
photon tenant create plato
photon tenant quota set plato
         --limits 'vm.count 100 COUNT, vm.memory 1000 GB, vm.cpu 500 COUNT'
photon project create --tenant "plato" --name "plato-prjt"
--limits "vm.memory 100 GB, vm.cpu 100 COUNT, vm 100 COUNT,
          persistent-disk 100 COUNT, persistent-disk.capacity 200 GB"
photon project quota set projectid1
--limits 'vm.count 100 COUNT, vmMemory 1000 GB, vmCpu 500 COUNT'
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 peristent 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 service 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 service` commands:

```
photon service list
```

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

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

See all the VMs in a cluster:

```
photon service list-vms <cluster-ID>
```

Delete it:

```
photon service 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
```

Sample output for a domain named `photon`:

```
    Login Access Token:
    Subject: faulkner@photon
    Groups: photongroup11, photonEveryone
    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 to work with ESXi hosts. You can, for example, add ESXi hosts to your Photon Controller cluster. 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

Connect to the Photon Platform installer VM as root by using SSH; example:

```
ssh root@198.51.100.212
```

(The default password for the installer VM, if you haven't changed it, is `changeme`.)

The Photon Platform installer VM includes a command-line utility that you use to install the Photon Platform VIBs on an ESXi host that you want to add to a Photon cluster. Here's the help for the command to install the VIBS on an ESXi host so that you can add it:

```
/opt/vmware/photon/controller/bin/photon-setup agent help
Usage: photon-setup agent <command> <arguments>
Command:
    install
    help
agent install:
    -esxi_host      <ESXi hostname or IP Address>
    -esxi_password  <Password to ESXi host>
    -lw_domain      <Lightwave domain name>
    -lw_password    <Lightwave domain Administrator password>
    -lw_dns         <Lightwave preferred DNS>
```

```
    -syslog_server  <Syslog endpoint>
```

Here's an example of the command to install the VIBs on an ESXi host:

```
/opt/vmware/photon/controller/bin/photon-setup agent
-esxi_host 192.0.2.23
-esxi_password  Secret1!
-lw_domain      lightwave-1.example.com
-lw_password    TopSecret!
-lw_dns         198.51.100.1
```

After you install the VIBs, you can terminate your SSH connection to the installer VM and use the Photon CLI on your workstation to add the ESXi host to the cluster 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:

```
photon host create -h
NAME:
   photon host create - Add a new host

USAGE:
   photon host create [command options]

DESCRIPTION:
   Add a new host to Photon Controller. You must a system administrator to add a host.

OPTIONS:
   --username value, -u value           username to create host
   --password value, -p value           password to create host
   --address value, -i value            ip address of the host
   --availability_zone value, -z value  availability zone of the host
   --tag value, -t value                tag for the host
   --metadata value, -m value           metadata for the host
```

After you create the host, you must provision it with the `photon host provision` command so that it is in a ready state:

```
photon host provision -h
NAME:
   photon host provision - Provision host with given id

USAGE:
   photon host provision <host-id>

DESCRIPTION:
   Provision a host given its id. You must be a system administrator to do this.
   This will configure photon controller agent and make the host ready.
```

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, Quotas, and Projects

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

5.1.1 Tenants

The top-level tenancy element in Photon Controller is called a tenant. A tenant has quotas 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 Quotas

Quotas represent allocations of physical or virtual objects. You allocate quotas under a tenant by using the same free-form syntax as flavor definitions.

Before you create a project, the tenant must have at least one quota. You can then create quotas for the tenant's projects.

For example, a quota 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. You can, however, change a quota after you set it. System administrators can resize tenant quotas, and tenant administrators can resize project quotas.

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

There is a strong relationship between flavors and quotas. 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 quota 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 quota, a user could create 50 VMs (since each uses two CPUs). Because the quota 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 quota at creation time. For instance, if a quota allows 100 virtual CPUs, a project could allow anywhere between 0 and 100 CPUs from the quota. Quotas, in other words, can be subdivided among projects.

5.2 Creating Tenants, Projects, and Quotas

Got a brand new installation of Photon Controller and not sure what to do next? The following examples demonstrate how to start working with Photon Controller from scratch by creating a tenant, a project, and a quota.

The commands assume that you have installed Photon Controller, have installed the Photon Controller CLI on a Linux workstation, and have access to the Photon Controller YAML file that you used to deploy Photon Controller. You can use the values in the YAML deployment file to replace the user names, passwords, and other variables that appear in curly brackets.

5.2.1 Set the Target to the Load Balancer

Get the IP address of the load balancer from your Photon Controller YAML deployment file and use it to set the target of the Photon Controller CLI on your workstation:

```
photon target set -c https://{load-balancer-ip}:443
```

5.2.2 Login

Log in as a Photon Controller administrator so that you can create tenants.

```
photon target login --username {lightwave-user-name}@{lightware-domain}
                  --password '{lightwave-user-password}'
```

5.2.3 Create Tenant

Create a new tenant with an allocation of resources—that is, a quota:

```
    photon tenant create 'Tom' -l 'vm 2000 COUNT, vm.cost 2000 COUNT,
  vm.cpu 2000 COUNT, vm.memory 2000 GB, ephemeral-disk 2000 COUNT,
  ephemeral-disk.capacity 2000 GB, ephemeral-disk.cost 2000 GB,
  persistent-disk 2000 COUNT, persistent-disk.capacity 2000 GB,
  persistent-disk.cost 2000 GB, storage.LOCAL_VMFS 2000 COUNT,
  storage.VSAN 2000 COUNT, sdn.floatingip.size 2000 COUNT'
    Comma-separated security group names, or hit enter for no security groups):
    CREATE_TENANT completed for 'tenant' entity cedc852f-f403-4cb0-9d7f-2614329a4229
```

Even though the term `quota` does not appear in the preceding command, the resource counts that the command allocates are the quota for this tenant.

5.2.4 List Tenants

```
photon tenant list
```

You should see the one you just created.

5.2.5 Set Tenant

Set the tenant to Tom:

```
photon tenant set Tom
```

5.2.6 Create a Project with a Quote

Create a project with quota. The project quota must be a subset of its tenant quota.

```
 photon project create TomProj1
--tenant Tom
--limits 'vm 100 COUNT, vm.memory 1000 GB, vm.cpu 500 COUNT'
```

5.2.7 List the Projects

```
photon project list
ID                                    Name      Limit               Usage
c6955c9d-ba88-4555-a2ad-8cd93c15d4fc  TomProj1  vm 100 COUNT       vm 0 COUNT
                                                vm.cpu 500 COUNT   vm.cpu 0 COUNT
                                                vm.memory 1000 GB  vm.memory 0 GB
```

5.2.8 Create Another Project Using 30 Percent of the Quota

Next, create a second project. This project uses 30 percent of its tenant's quota.

```
photon project create TomProj2 --tenant Tom --percent 30

Tenant name: Tom
Creating project name: TomProj2

Please make sure limits below are correct:
1: subdivide.percent, 0.3, COUNT
Are you sure [y/n]? y
CREATE_PROJECT completed for 'project' entity 802798a0-07b7-4fc5-b645-bf0c6212e62d
```

5.2.9 List the Projects Again

Now there should be two of projects:

```
photon project list
ID          Name      Limit                                Usage
80279...    TomProj2  storage.LOCAL_VMFS 6000 COUNT     storage.LOCAL_VMFS 0 COUNT
                      storage.VSAN 6000 COUNT           storage.VSAN 0 COUNT
                      vm 6000 COUNT                     vm 0 COUNT
                      vm.memory 6000 GB                 vm.memory 0 GB
                      persistent-disk 6000 COUNT        persistent-disk 0 COUNT
                      persistent-disk.capacity 6000 GB  persistent-disk.capacity 0 GB
                      ephemeral-disk.cost 6000 GB       ephemeral-disk.cost 0 GB
                      persistent-disk.cost 6000 GB      persistent-disk.cost 0 GB
                      sdn.floatingip.size 6000 COUNT    sdn.floatingip.size 0 COUNT
                      vm.cost 6000 COUNT                vm.cost 0 COUNT
                      vm.cpu 6000 COUNT                 vm.cpu 0 COUNT
```

```
                      ephemeral-disk 6000 COUNT          ephemeral-disk 0 COUNT
                      ephemeral-disk.capacity 6000 GB    ephemeral-disk.capacity 0 GB
c6955...    TomProj1  vm 100 COUNT                       vm 0 COUNT
                      vm.cpu 500 COUNT                   vm.cpu 0 COUNT
                      vm.memory 1000 GB                  vm.memory 0 GB
```

Be careful not to make your quotas too big, or bigger than they need to be, to avoid getting a quota error.

When you create a VM, the "vm.memory" and "vm.cpu" are required keys for compute. For storage, you must specify ephemeral-disk and ephemeral-disk.capacity if you created a persistent disk. Other quota items are optional.

5.2.10 Show Tenant Quota

Now, take a look at the quota for the tenant:

```
photon tenant quota show Tom
Limits:
  storage.VSAN              2000  COUNT
  ephemeral-disk.capacity   2000  GB
  persistent-disk.capacity  2000  GB
  vm.cost                   2000  COUNT
  vm.memory                 2000  GB
  ephemeral-disk            2000  COUNT
  persistent-disk.cost      2000  GB
  persistent-disk           2000  COUNT
  vm.cpu                    2000  COUNT
  ephemeral-disk.cost       2000  GB
  storage.LOCAL_VMFS        2000  COUNT
  sdn.floatingip.size       2000  COUNT
  vm                        2000  COUNT
Usage:
  persistent-disk.capacity  6000  GB
  vm.cost                   6000  COUNT
  vm.memory                 7000  GB
  ephemeral-disk            6000  COUNT
  storage.VSAN              6000  COUNT
  ephemeral-disk.capacity   6000  GB
  ephemeral-disk.cost       6000  GB
  storage.LOCAL_VMFS        6000  COUNT
  sdn.floatingip.size       6000  COUNT
  vm                        6100  COUNT
  persistent-disk.cost      6000  GB
  persistent-disk           6000  COUNT
  vm.cpu                    6500  COUNT
```

5.2.11 Show Project Quota

And drill down into the one of the project's quotas by using the project's ID:

```
photon project quota show c6955c9d-ba88-4555-a2ad-8cd93c15d4fc
Limits:
  vm.cpu     500   COUNT
  vm.memory  1000  GB
  vm         100   COUNT
```

```
Usage:
  vm.cpu     0   COUNT
  vm.memory  0   GB
  vm         0   COUNT
```

5.2.12 Updating Quotas

If you forgot to provide a quota item or didn't provide enough resources when you created a tenant or project, you can use the `quota update` command to make adjustments.

Note that you might get an error when updating a quota if you attempt to lower a quota limit below its current usage.

5.2.12.1 Update a Tenant's Quota

For example, here's how to add a missing vm.count to a quota:

```
photon tenant quota update Tom -l 'vm.count 2000 COUNT'
Tenant name: Tom
Please make sure limits below are correct:
1: vm.count, 2000, COUNT
Are you sure [y/n]? y
UPDATE_QUOTA completed for 'tenant' entity cedc852f-f403-4cb0-9d7f-2614329a4229
```

5.2.13 Update a Project Quota

You can also update a project's quota. As an example, this command increases the vm.cpu quota to 1000:

```
photon project quota update c6955c9d-ba88-4555-a2ad-8cd93c15d4fc -l 'vm.cpu 1000 COUNT'
Project Id: c6955c9d-ba88-4555-a2ad-8cd93c15d4fc
Please make sure limits below are correct:
1: vm.cpu, 1000, COUNT
Are you sure [y/n]? y
UPDATE_QUOTA completed for 'project' entity c6955c9d-ba88-4555-a2ad-8cd93c15d4fc
```

Verify your change:

```
photon project quota show c6955c9d-ba88-4555-a2ad-8cd93c15d4fc
Limits:
  vm.count   200   COUNT
  vm.cpu     1000  COUNT
  vm.memory  1000  GB
  vm         100   COUNT
Usage:
  vm.count   0   COUNT
  vm.cpu     0   COUNT
  vm.memory  0   GB
  vm         0   COUNT
```

5.2.14 Removing Entries from a Quota

Here's how to exclude a quota entry for a project's quota:

```
photon project quota exclude c6955c9d-ba88-4555-a2ad-8cd93c15d4fc -l 'vm.count 200 COUNT'

Project Id: c6955c9d-ba88-4555-a2ad-8cd93c15d4fc
Please make sure limits below are correct:
1: vm.count, 200, COUNT
Are you sure [y/n]? y
DELETE_QUOTA completed for 'project' entity c6955c9d-ba88-4555-a2ad-8cd93c15d4fc
```

And here's how to do it for a tenant's quota:

```
photon tenant quota exclude Tom -l 'vm.count 2000 COUNT'

Tenant name: Tom
Please make sure limits below are correct:
1: vm.count, 2000, COUNT
Are you sure [y/n]? y
DELETE_QUOTA completed for 'tenant' entity cedc852f-f403-4cb0-9d7f-2614329a4229
```

Keep in mind that a project quota consumes part of a tenant's quota. When removing a quota item, remember to remove the consumer first.

5.2.15 Setting Quota

Setting a quota totally wipes out the existing quota and replaces it with the new quota:

```
photon project quota set c6955c9d-ba88-4555-a2ad-8cd93c15d4fc -l 'vm 500 COUNT'

Project Id: c6955c9d-ba88-4555-a2ad-8cd93c15d4fc
Please make sure limits below are correct:
1: vm, 500, COUNT
Are you sure [y/n]? y
RESET_QUOTA completed for 'project' entity c6955c9d-ba88-4555-a2ad-8cd93c15d4fc
```

Check your work:

```
photon project quota show c6955c9d-ba88-4555-a2ad-8cd93c15d4fc
Limits:
  vm  500  COUNT
Usage:
  vm  0   COUNT
```

Now that you've created a tenant, a project, and a quota, move on to creating a VM.

5.3 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 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, quotas, 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.

5.3.1 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>
```

5.3.2 Commands for Working with Tenants

List tenants:

```
photon tenant list
```

Show information about a tenant:

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

Show the limits and project allocations of a quota:

```
photon tenant quota show <tenant-name>
```

List projects:

```
photon project list
```

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

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

5.3.3 Related API Endpoints

```
/tenants
/projects
```

5.4 Setting Up a Project

A project carves out a set of resources for a tenant. A project requires a quota, and the project can consume all or part of the resources defined by its quota.

To create a project, you must be logged in as a system administrator.

5.4.1 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
NAME:
   photon project create - Create a new project

USAGE:
   photon project create [command options] <project-name>

DESCRIPTION:
   Create a new project within a tenant and assigns some or all of its tenant quota.
   Only system administrators can create new projects.
   If default-router-private-ip-cidr option is omitted,
   it will use 192.168.0.0/16 as default router's private IP CIDR.
   A quota for the project can be defined during project creation and
   it is defined by a set of maximum resource costs. Each usage has a type,
   a numnber (e.g. 1) and a unit (e.g. GB). You must specify at least one cost
   Valid units:  GB, MB, KB, B, or COUNT
   Common costs:
     vm.count:            Total number of VMs (use with COUNT)
     vm.cpu:              Total number of vCPUs for a VM (use with COUNT)
     vm.memory:           Total amount of RAM for a VM (use with GB, MB, KB, or B)
     disk.capacity:       Total disk capacity (use with GB, MB, KB, or B)
     disk.count:          Number of disks (use with COUNT)
     sdn.floatingip.size: Number of floating ip

OPTIONS:
   --limits value, -l value
     Project limits (key value unit)
   --percent value, -p value
     Project limits (0 to 100 percent of tenant quota) (default: 0)
   --tenant value, -t value
     Tenant name for project
   --security-groups value, -g value
     Security Groups for project
   --default-router-private-ip-cidr value, -c value
     Private IP range of the default router in CIDR format.
     Default value: 192.168.0.0/16
```

5.4.2 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 by setting a quota, and creates a project that can use use all the resources in the pool.

```
photon tenant create plato
photon tenant quota set plato
   --limits 'vm.count 100 COUNT, vm.memory 1000 GB, vm.cpu 500 COUNT'
photon project create --tenant "plato" --name "plato-prjt"
   --limits "vm.memory 100 GB, vm.cpu 100 COUNT, vm 100 COUNT,
   persistent-disk 100 COUNT, persistent-disk.capacity 200 GB"
```

When you create a project, you can optionally set the project's quota to use a percent of its tenant quota. The following command, for example, sets the project quota to 30 percent of its tenant quota:

```
photon project create project2 --tenant tenant1 --percent 30
```

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.

5.5 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.

5.5.1 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.

5.5.2 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
```

5.5.3 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.

5.5.4 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.

5.5.5 Deleting Images

You can delete an image like this:

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

5.6 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.

5.6.1 Creating Images

Here's how to create an image:

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

5.6.2 Viewing the List of Images

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

```
photon image list
```

5.6.3 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
```

5.6.4 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
```

5.6.5 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
```

5.7 Replicating Images in Datastores

An image datastores is an ESXi datastore that is typically shared between multiple ESXi hosts. The image datastore holds the images for creating virtual machines, Kubernetes clusters, and other resources.

5.7.1 Image Replication and Datastores

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

When you create an image in Photon Controller, you mark it as either `ON_DEMAND` or `EAGER`. Both types are 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 approach 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. A datastore specified as `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 uses all the datastore that it finds. Images will only be replicated or copied to the set of allowed datastores.

Datastores attached after the system is installed will be found and will be used if they are allowed.

It is acceptable to have only one datastore in your deployment as along as it is shared across all hosts. A single, shared datastore 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.

5.8 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 quota you are using.

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

5.8.1 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.

5.8.2 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 quota.
- 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.

5.8.3 Creating Flavors

You create flavors by using the `photon flavor create` command. You'll be prompted for a few details. Here's an example:

```
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
```

The complete help output for the command contains additional examples:

```
photon flavor create -h
NAME:
   photon flavor create - Create a flavor
USAGE:
   photon flavor create [command options]
DESCRIPTION:
   This creates a new flavor. Only system administrators can create flavors.
   A flavor is defined by a set of costs. Each cost has a type (e.g. vm.memory),
   a numnber (e.g. 1) and a unit (e.g. GB). VM flavors must specify at
   least two costs: vm.memory and vm.cpu. Disk flavors must specify at
   least one cost, and it's typically the number of disk (typically one)
   Valid units:  GB, MB, KB, B, or COUNT
   Common costs:
     vm.count:               Number of VMs this is (probably 1 COUNT)
     vm.cpu:                 Number of vCPUs for a VM (use with COUNT)
     vm.memory:              Amount of RAM for a VM (use with GB, MB, KB, or B)
     persistent-disk.count:  Number of persistent disk (probably 1 COUNT)
     ephemeral-disk.count:   Number of ephemeral disks (probably 1 COUNT)
   Example VM flavor command:
      photon flavor create --name f1 --kind vm
                           --cost 'vm.memory 1 GB, vm.cpu 1 COUNT'
   Example disk flavor:
      photon flavor create --name f1 --kind persistent-disk.count
                           --cost 'persistent-disk.count 1 COUNT'
OPTIONS:
   --name value, -n value  Flavor name
   --kind value, -k value  Flavor kind: persistent-disk, ephemeral-disk, or vm
   --cost value, -c value  Comma-separated costs. Each cost is "type number unit"
```

5.8.4 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
```

5.8.5 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
```

5.9 Provisioning Virtual Machines

5.9.1 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
```

5.9.2 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
```

5.9.3 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
```

5.9.4 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
```

5.9.5 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
```

5.9.6 Other Options for Creating VMs

You can run the `photon vm create` command with additional options, the following help output for the command shows:

```
photon vm create --help
NAME:
   photon vm create - Create a new VM

USAGE:
   photon vm create [command options]

OPTIONS:
   --name value, -n value              VM name
   --flavor value, -f value            VM flavor
   --image value, -i value             Image ID
   --boot-disk-flavor value, -b value  Boot disk flavor
   --disks value, -d value             VM disks
   --environment value, -e value       VM environment({key:value})
   --affinities value, -a value        VM Locality(kind id)
   --networks value, -w value          VM Networks(id1, id2)
   --tenant value, -t value            Tenant name
   --project value, -p value           Project name
```

For example, adding the `--boot-disk-flavor` option to the `photon vm create` command inserts the boot disk as the first disk implicitly. In interactive mode, the Photon CLI may prompt you for additional data disks, which can no longer be a boot disk. When `--boot-disk-flavor` does not appear in the command in interactive mode, you may be prompted to supply it but command prompt accepts an empty value. If it's empty, you are prompted to supply disk and at least one boot disk is required.

5.10 Creating a Subnet

Here's how you can use the `photon subnet` command to work with subnets, including virtualized subnets powered by VMware NSX, on the ESXi hosts in a Photon Controller cluster.

You can use either NSX virtualized networking or traditional networking; you cannot mix them. If you want to use NSX, you must set up an NSX network before you install Photon Controller. You cannot change the type of networking after you install Photon Controller. See Setting Up NSX. To take full advantage of the power of Photon Platform, it is recommended that you use Photon Controller with NSX.

To support provider networks, Photon lets you provide NAT subnets.

As a project user in Photon Controller, you can also do the following when you are using NSX virtualized networking:

- Create additional NSX Tier-1 routers for your projects with the CLI, empowering your projects with flexibility.
- Create multiple subnets under Tier-1 routers.
- Associate workloads with each subnet; but Photon Controller supports only 1 subnet per workload.

- Associate with a workload a floating IP address from a global IP address pool.
- Disassociate a floating IP address from a workload and return it to the global pool.

With the Photon command-line utility on your workstation, you can view the options of the `photon subnet` command like this:

```
photon subnet -h
NAME:
   photon subnet - options for subnet
USAGE:
   photon subnet command [command options] [arguments...]
COMMANDS:
     create       Create a new subnet
     delete       Delete subnet with specified id
     list         List all subnets.
     show         Show subnet info with specified id
     update       Update subnet
     set-default  Set default subnet
OPTIONS:
   --help, -h  show help
```

5.10.1 Networks in the Context of Multitenancy

In Photon Controller, there can be multiple tenants. Each tenant can have multiple projects, and each project can have multiple VMs. A project can have multiple subnets, including a default subnet. A virtual machine can be associated with a subnet when you create the VM; otherwise, the VM is associated with the default subnet.

5.10.2 Checking the Default Subnet

With a typical installation of Photon Controller without NSX, there is one default subnet, usually `VM Network`. You can check for a default subnet by running the following command:

```
photon subnet list
ID                    Name        State  PortGroups    Descriptions         IsDefault
f858e2fd5437c42096940  vm-network  READY  [VM Network]  photon subnet list  true
```

```
Total: 1
```

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

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

For an installation of Photon Controller with NSX, you can define a default subnet per project under the default Tier-1 router.

5.10.3 Creating a New Subnet

Adding a new subnet 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 subnet in the Photon Platform web interface, on the `Cloud` page, click `Networks` and then click `New Network`:

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

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

```
photon subnet create --help
NAME:
   photon subnet create - Create a new subnet
USAGE:
   photon subnet create [command options]
DESCRIPTION:
   Create a new subnet.
   Examples:
   Virtual Subnet:
   photon subnet create -n subnet-1 -d test-subnet -i 192.168.0.0/16 -r 5f8cap789
   Physical Subnet:
   photon subnet create -n subnet-1 -d test-subnet -p port1,port2
OPTIONS:
   --name value, -n value           Name of the subnet
   --description value, -d value    Description of the subnet
   --privateIpCidr value, -i value  The private IP range of subnet in CIDR format,
                                    e.g.: 192.168.0.0/16
   --router value, -r value         The id of the router on which subnet is to be created
   --type value, -t value           Type of subnet to be created.
                                    Types: NAT, NO_NAT or PROVIDER. Default: NAT
   --portgroups value, -p value     PortGroups associated with subnet
                                    (only for physical subnet)
```

5.10.4 Setting the Default Subnet

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

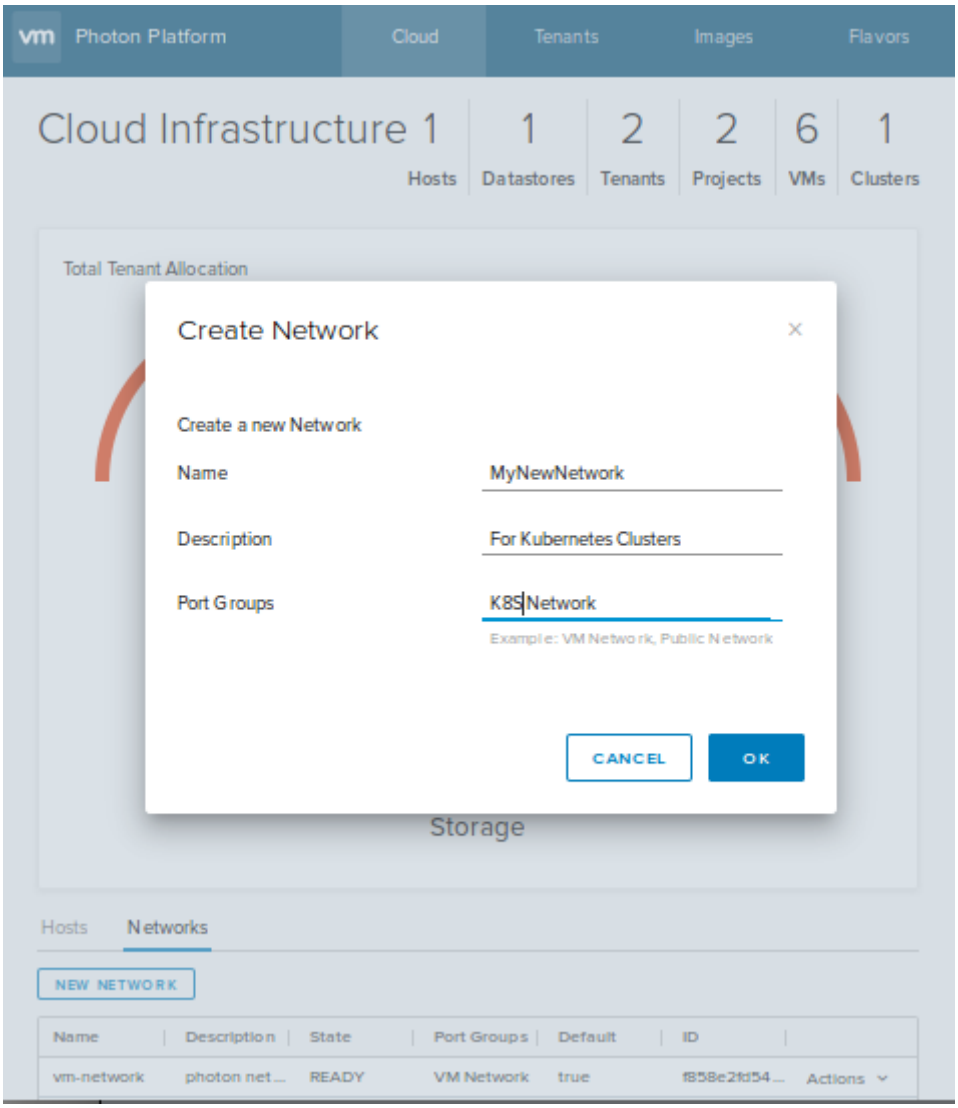

Figure 2: Adding a Network

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

5.11 Setting Up Availability Zones

Availability zones group ESXi hosts to isolate an application running on virtual machines, to provide data locality or physical affinity, to improve performance, or to ensure high availability.

To group ESXi hosts into an availability zone, you first create a zone with the `photon availability-zone create` command and then add ESXi hosts to the zone. After the zone is established with a set of ESXi hosts, you create VMs and assign them to the zone.

A set of VMs running an application can be segmented into an availability zone to ensure that the VMs reside on the same physical rack or behind a switch for low latency. Inversely, you can place VMs on separate physical infrastructure to provide high availability. Similarly, availability zones can locate VMs on the hardware that provides the right level of performance for the application.

The following diagram illustrates the role of availability zones in the context of Photon Platform's multitenancy model:

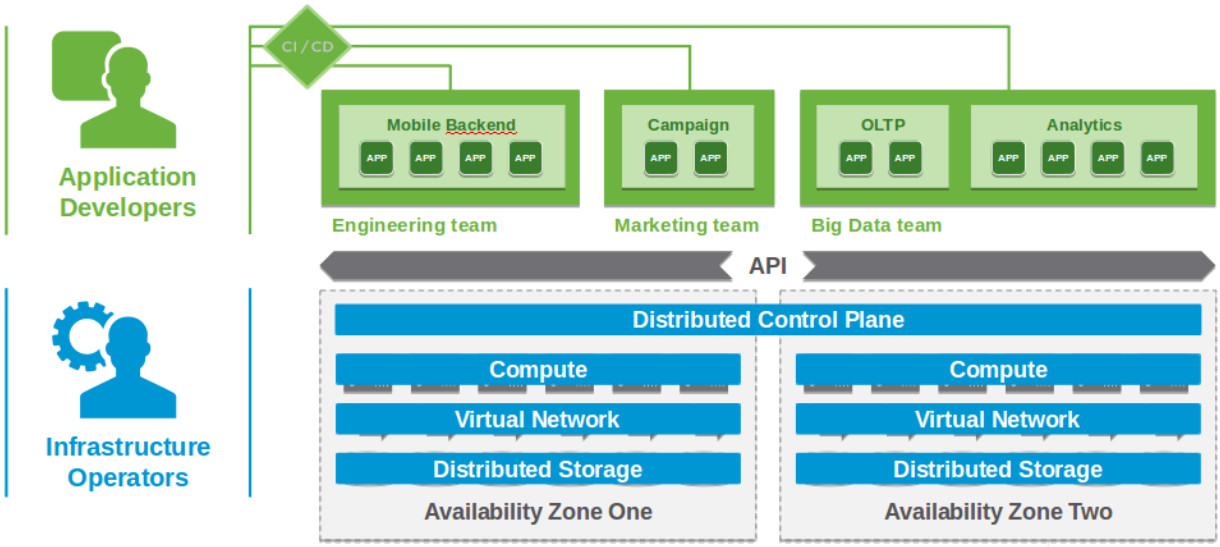

Figure 3: Availability Zones

Keep in mind, though, that availability zones do not correspond to tenants: Different tenants can share the same availability zone.

5.12 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.

5.12.1 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:

`https://<ip-address-of-load-balancer>:4343`

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

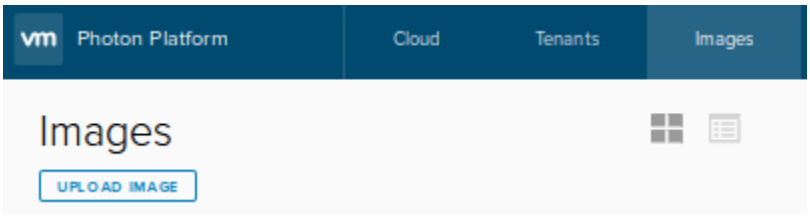

Figure 4: Upload Image

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`:

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:

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.

5.12.2 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
```

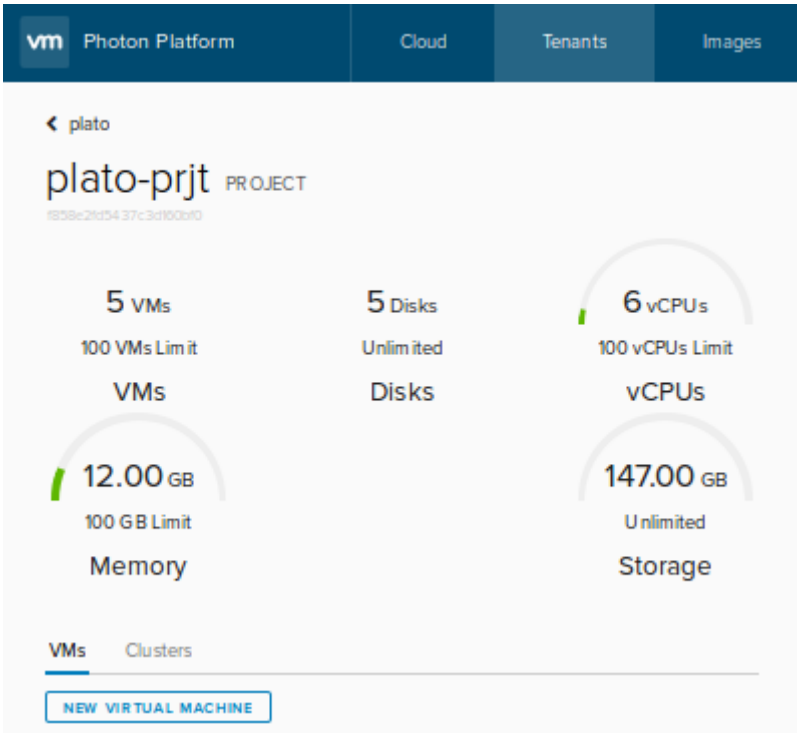

Figure 5: New Virtual Machine

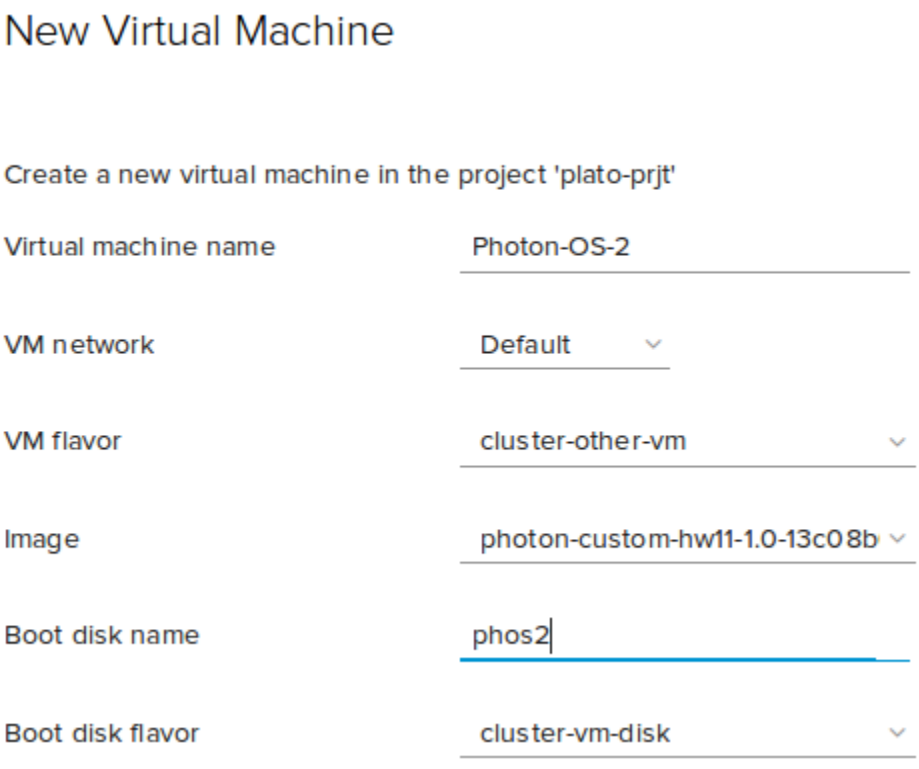

Figure 6: The Form for a New VM

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.

5.13 Photon OS Documentation

Photon OS is an open-source, minimalist Linux container host from VMware optimized for VMware vSphere deployments and cloud-native applications. On Photon Platform, Photon OS is available on demand to run Docker containers.

Here's a list of key documents and GitHub pages on Photon OS. Additional material resides in the Photon OS GitHub repository.

- Download Photon OS 1.0
- Installation and Administration Guide
- Troubleshooting Guide
- FAQ
- Photon OS GitHub Wiki
- Getting Started Guides
  - Quick Start
  - Running Photon OS on vSphere
  - Running Photon OS on Fusion
  - Running Photon OS on vCloud Air
  - Running Photon OS on Google Compute Engine
  - Running Photon OS on Amazon Elastic Cloud Compute
- Guides
  - Installation and Administration Guide
  - Troubleshooting Guide
  - Photon RPM OSTree
- How-To Articles
  - Setting Up a Swarm Cluster with DNS
  - Setting Up a Mesos Cluster
  - Setting Up Marathon for a Mesos Cluster
  - Setting Up DCOS CLI for Mesos
  - Setting Up Mesos DNS on a Mesos Cluster
  - Setting Up a Network PXE Boot Server
  - Working with Kickstart
  - Running Kubernetes
  - Running Rocket Containers
  - Mounting Remote File Systems
  - Building Photon OS from the Source Code
- Security
  - Security Advisories

6 Deploying Clusters

6.1 Creating a Kubernetes Cluster

Table of Contents

6.1.1 Introduction

Setting up a Kubernetes cluster to run a web application demonstrates the power of Kubernetes as a service. Photon Platform supports Kubernetes clusters with multiple masters, persistent volumes, and authentication. A Kubernetes cluster can use traditional networking or virtualized networking powered by VMware NSX.

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

- Images
- Virtual machines
- Disks
- Quotas

6.1.2 Requirements

A Kubernetes cluster carries several requirements:

- A static IP address to assign as the master IP address of the cluster.
- A static IP address for the load balancer.
- 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 without VMware NSX. 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 tenant create --help
photon service create -h
```

The VMware NSX option: If you deployed Photon Platform with NSX, you can set up a Kubernetes cluster with a virtualized network; see Setting Up a Kubernetes Cluster with NSX.

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, build, and other numbers might be slightly different:

```
kubernetes-1.6.0-pc-1.2-5caa892.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-1.6.0-pc-1.2-5caa892.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, sets a quota to give it a pool of resources, and creates a project that uses use all the resources in the pool.

```
photon tenant create plato
photon tenant quota set plato
         --limits 'vm.count 100 COUNT, vm.memory 1000 GB, vm.cpu 500 COUNT'
photon project create --tenant "plato" --name "plato-prjt"
    --limits "vm.memory 100 GB, vm.cpu 100 COUNT, vm 100 COUNT,
    persistent-disk 100 COUNT, persistent-disk.capacity 200 GB,
    ephereral-disk 100 COUNT, ephereral-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. This flavor is for etcd nodes and Kubernetes workers.
- cluster-vm-disk. This flavor is the default ephemeral disk for a Kubernetes cluster.

When you create a Kubernetes cluster, Photon Controller uses 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 subnet create --name "vm-network" --portgroups "VM Network"
```

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

```
photon subnet 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 service create -n kube-socrates -k KUBERNETES --master-ip 203.0.113.208
    --load-balancer-ip 203.0.113.207
    --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 `service 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.

6.1.8.1 The `photon service create` Command

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

```
photon service create -h
NAME:
   photon service create - Create a new service
USAGE:
   photon service create [command options]
DESCRIPTION:
   Create a new Kubernetes service or Harbor Docker registry.
   Example:
   photon service create -n k8-service -k KUBERNETES --dns 10.0.0.1
     --gateway 192.0.2.1 --netmask 255.255.255.0 --master-ip 192.0.2.20
     --container-network 10.2.0.0/16 --etcd1 192.0.2.21
     -c 1 -v cluster-vm -d small-disk --ssh-key ~/.ssh/id_dsa.pub
OPTIONS:
   --tenant value, -t value            Tenant name
   --project value, -p value           Project name
   --name value, -n value              Service name
   --type value, -k value              Service type (KUBERNETES or HARBOR)
   --vm_flavor value, -v value         VM flavor name for master and worker
   --master-vm-flavor value, -m value  Override master VM flavor
   --worker-vm-flavor value, -W value  Override worker VM flavor
   --disk_flavor value, -d value       Disk flavor name
   --subnet_id value, -w value         VM subnet ID
   --image-id value, -i value          Image ID
   --worker_count value, -c value      Worker count (default: 0)
   --dns value                         VM network DNS server IP address
   --gateway value                     VM network gateway IP address
   --netmask value                     VM network netmask
   --number-of-masters value           Number of Kubernetes masters
     (required for Kubernetes services with virtual networking) (default: 0)
   --master-ip value                   Kubernetes master IP address
     (required for Kubernetes services with physical networking)
   --master-ip2 value                  Static IP address with which
                                       to create Kubernetes node 2
   --load-balancer-ip value            Kubernetes load balancer IP address
     (required for Kubernetes services with physical networking)
   --container-network value           CIDR representation of the container network,
     e.g. '10.2.0.0/16' (required for Kubernetes services)
   --number-of-etcds value             Number of Etcd instances for Kubernetes
     (required for Kubernetes services with virtual networking) (default: 0)
   --etcd1 value                       Static IP address with which to create etcd node 1
     (required for Kubernetes services with physical networking)
   --etcd2 value                       Static IP address with which to create etcd node 2
   --etcd3 value                       Static IP address with which to create etcd node 3
   --ssh-key value                     The file path of the SSH key
   --registry-ca-cert value            The file path of the file containing the CA
     certificate for a docker registry (optional)
   --admin-password value              The Harbor registry admin password (optional).
     The password needs to have at least 7 characters with 1 lowercase letter, 1 capital
     letter and 1 numeric character. If not specified, the default user name is admin
     and the password is Harbor12345
   --batchSize value                   Batch size for expanding worker nodes (default: 0)
   --wait-for-ready                    Wait synchronously for the service to become
                                       ready and expanded fully
```

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:

`https://<ip-address-of-photon-load-balancer>:4343`

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

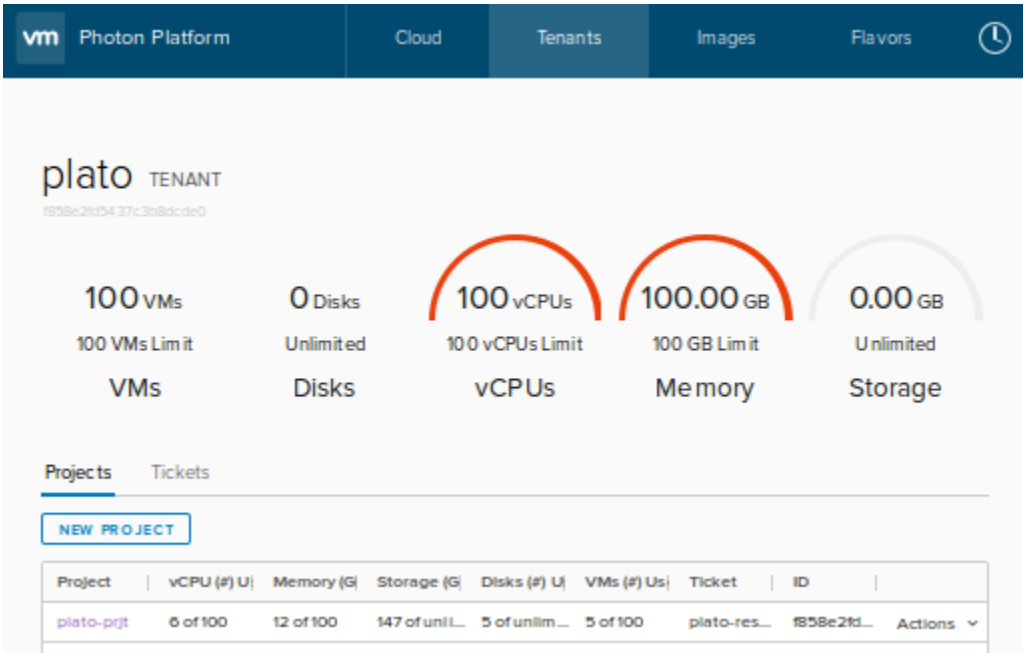

Figure 7: A Tenant in Photon Platform

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

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

6.1.10 Opening the Kubernetes Dashboard

To open the Kubernetes web interface in a browser, you first must obtain information about authentication for the Kubernetes cluster from Photon Controller and then you must add it to your kubectl configuration file. Here's how:

First, get the kubectl authentication information from Photon Controller by running the following command on your workstation. Replace the username and password with your Photon Controller username and password:

```
photon service get-kubectl-auth -u <username> -p <password> <service-id>
```

Here's and example:

```
photon service get-kubectl-auth -u administrator@example.com
        -p 'MySecret1!' 446c5839-992c-456a-b271-bc2aa48e88fc
```

Second, copy the command's output and add it to the following command to insert it into the kubectl configuration. Here's an example with sample output:

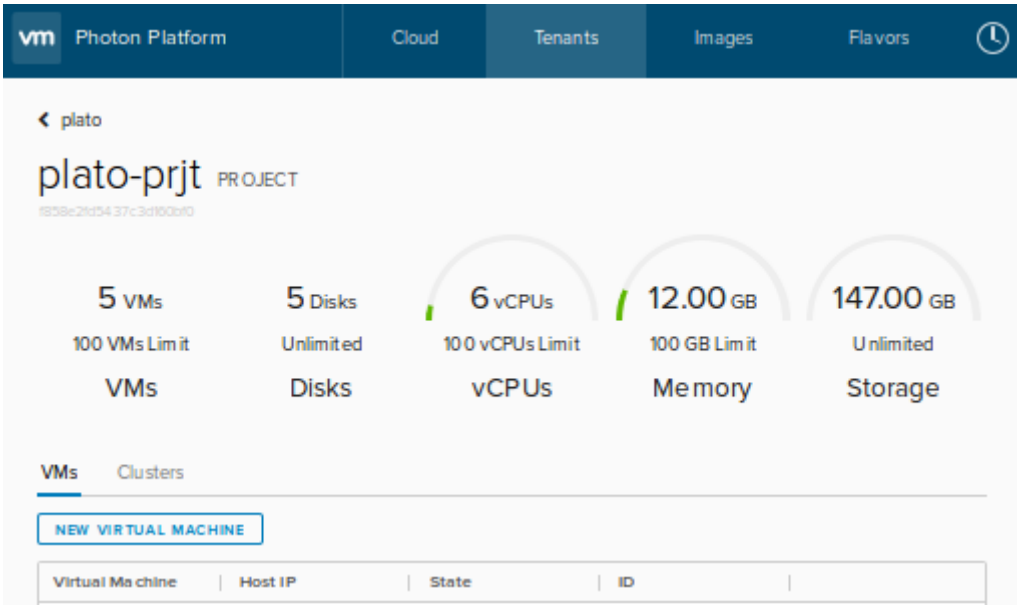

Figure 8: A Project in Photon Platform

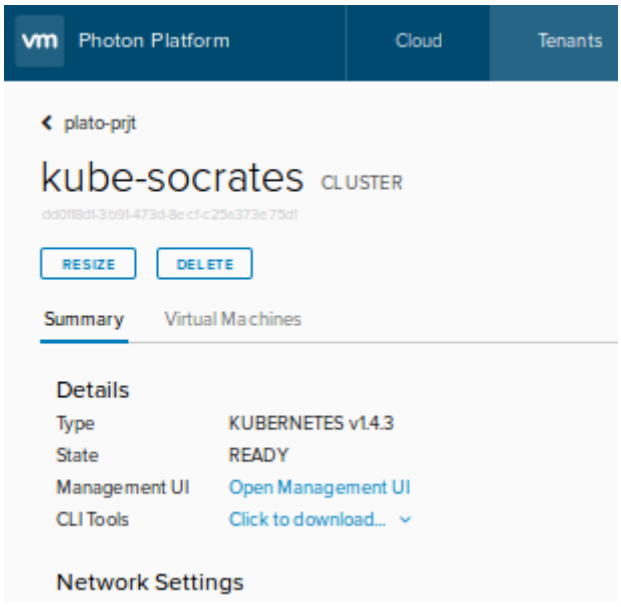

Figure 9: A Kubernetes Cluster

```
kubectl config set-credentials administrator@example.com
--auth-provider=oidc
--auth-provider-arg=idp-issuer-url=https://198.51.100.3/openidconnect/example.com
--auth-provider-arg=client-id=fc50e7cd-7166-4025-907b-c2af210fb902
--auth-provider-arg=client-secret=fc50e7cd-7166-4025-907b-c2af210fb902
--auth-provider-arg=refresh-token=eyJhbGciOiJSUzI1NiJ9.eyJzdWIi ...
--auth-provider-arg=id-token=eyJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJhZ ...
--auth-provider-arg=idp-certificate-authority=/tmp/lw-ca-cert-FNKP.pem
```

Next, set the cluster and server:

```
kubectl config set-cluster <cluster-name>
--server <k8s-load-balancer-address>
--insecure-skip-tls-verify=true
```

Example:

```
kubectl config set-cluster default-cluster
--server https://198.51.100.166:6443
--insecure-skip-tls-verify=true
```

Next, set the context in the kubectl config:

```
kubectl config set-context <context-name>
--cluster <cluster-name>
--user <username>
```

Example:

```
kubectl config set-context default-system
--cluster default-cluster
--user administrator@example.com
```

Now make the context the current context:

```
kubectl config use-context <context-name>
```

Example:

```
kubectl config use-context default-system
```

Verify that you set it up correctly and that it works:

```
kubectl get pods --all-namespaces
```

Finally, launch the Kubernetes user interface and go to it by entering the following URL in your web browser:

```
kubectl proxy
http://localhost:8001/ui
```

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, `kubectl`, which you can quickly download through the Photon Platform web interface, as the following image illustrates:

Or you can download the version of `kubectl` for a 64-bit Linux machine from the Kubernetes web site; see the Kubernetes user guide for instructions on how to download versions for other operating systems.

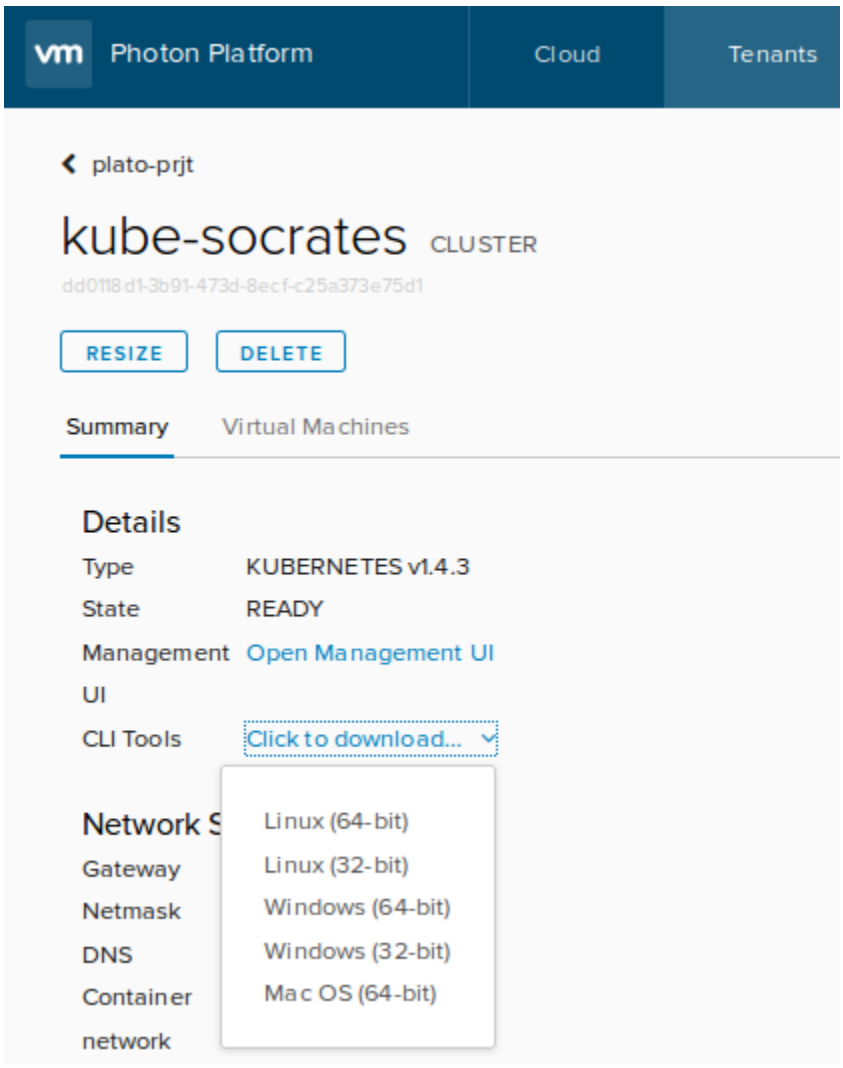

Figure 10: Install kubectl

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
```

Now you can 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:

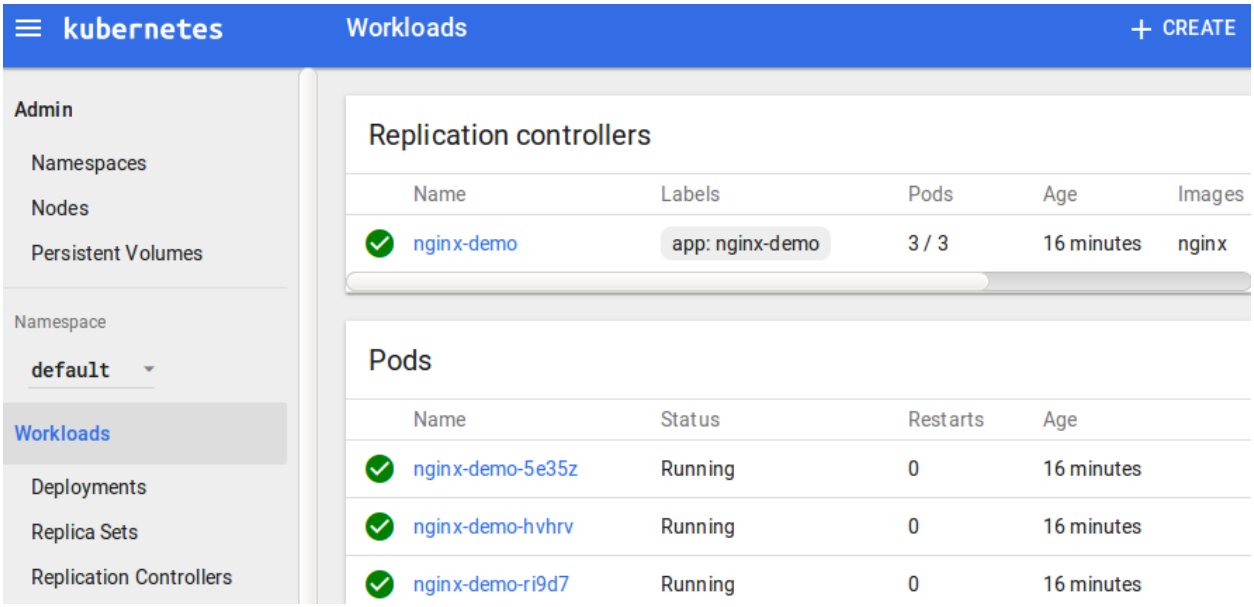

Figure 11: Kubernetes Workloads

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:

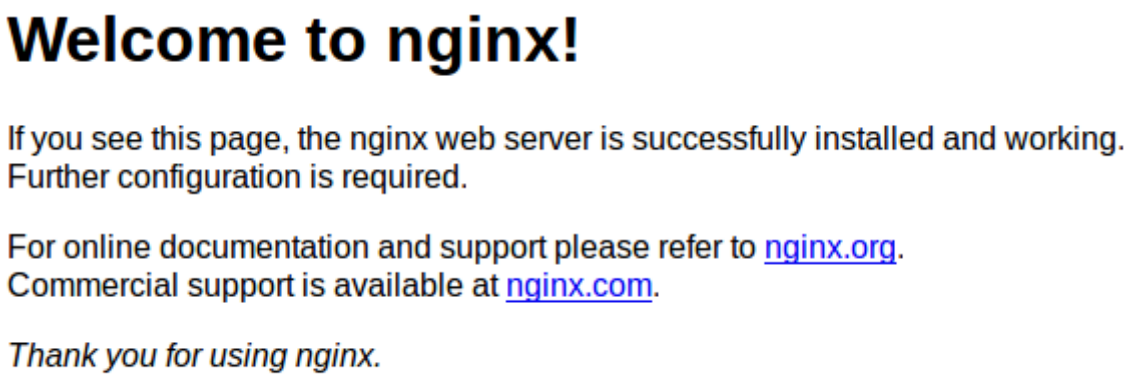

Figure 12: nginx Welcome Screen

6.1.12 Upgrading to New Version of Kubernetes

You can upgrade to a new version of Kubernetes by running the `photon cluster change-version` command. Here's the help output for the command:

```
photon cluster change-version -h
NAME:
   photon service change-version - Configure the service
   to use specified image id
USAGE:
   photon service change-version [command options] service-id
DESCRIPTION:
   Example: photon service change-version
   9b159e92-9495-49a4-af58-53ad4764f616
   -i 2aeaf034-3b02-4873-a6fc-f92615dca849
OPTIONS:
   --image-id value, -i value  Image ID
   --wait-for-ready            Wait synchronously for the
   service to become ready and fully upgraded
```

6.1.13 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 service list | awk '{print $1}' | xargs -n 1 photon -n service 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 service list` command; example:

```
photon service 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 service show` command; example with abridged output:

```
photon -n service 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:v1.6.0 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
```

6.1.13.1 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"
```

6.1.13.2 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.

6.1.13.3 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 service delete <cluster_ID>
```

6.1.13.4 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>
```

6.1.14 Related

- Setting Up a Kubernetes Cluster with NSX
- Deploying Tomcat on a Kubernetes Cluster
- Persistent Volumes for Kubernetes

6.2 Setting Up a Kubernetes Cluster with NSX

6.2.1 Introduction

Setting up a Kubernetes cluster powered by VMware NSX demonstrates the power of Kubernetes as a service with virtual networking.

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

- Images
- Virtual machines
- Disks
- Quotas
- Virtual routers, switches, and subnets

6.2.2 Requirements

A Kubernetes cluster powered by NSX carries several requirements:

- NSX is installed and configured to work with Photon Controller.
- The Kubernetes image for Photon Controller.
- Enough floating IP addresses for the master and etcd nodes.

The instructions assume that you are using a Linux workstation with the Photon Controller CLI connected to a deployment of Photon Controller on ESXi with VMware NSX. You can also use a Mac or Windows workstation with the Photon Controller CLI installed.

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

```
photon --help
photon service create -h
```

For more information about creating a Kubernetes cluster, or for instructions on creating a cluster without using NSX, see Creating a Kubernetes Cluster.

6.2.3 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 file; it has a file name that looks like this, though the version, build, and other numbers might be slightly different:

```
kubernetes-1.6.0-pc-1.2-5caa892.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 target login -u administrator@photon.local.v -p 'MySecret1!'
photon image create kubernetes-1.6.0-pc-1.2-5caa892.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. (Another option is to specify the image at the time of creating the cluster by using the `--image-id` option; see the help output for the `photon service create` command.)

```
photon image list
photon -n deployment enable-cluster-type default -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.

6.2.4 Creating a Tenant, a Project, and a Quota for the Cluster

A tenant is a unit of administrative control allocated a quota for projects, such as a Kubernetes cluster. The command creates a tenant and gives it quota of resources with the `limits` option. To provide the tenant with resources for NSX and Kubernetes clusters, the quota includes persistent disks, VMware vSAN storage, and floating IP addresses.

(This example code block requires that vSAN is installed and set up to work with Photon Controller. However, vSAN is not required for the Kubernetes cluster with NSX, and you can provision a different set of resources in the quota. See Creating a Tenant, a Project, and a Quota.)

```
photon tenant create demoTenant
--limits 'vm 10000 COUNT, vm.cost 20000 COUNT, vm.cpu 10000 COUNT,
vm.memory 10000 GB, ephemeral-disk.count 20000 COUNT,
ephemeral-disk.capacity 20000 GB, ephemeral-disk.cost 20000 GB,
ephemeral-disk 9999 COUNT, persistent-disk.count 20000 COUNT,
persistent-disk.capacity 20000 GB, persistent-disk.cost 20000 GB,
storage.LOCAL_VMFS 20000 COUNT, storage.VSAN 20000 COUNT,
sdn.floatingip.size 20000 COUNT'
```

The following command creates a project that uses 50 percent of the tenant's quota:

```
photon project create demoProject --tenant demoTenant --percent 50  -c 192.168.192.0/24
```

6.2.5 Creating Flavors for the Cluster

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.

The next two commands specify custom flavors. These custom flavors control the size of the resources that the cluster will use.

```
photon flavor create --name "small-vm" --kind "vm"
--cost "vm 1.0 COUNT, vm.cpu 3.0 COUNT, vm.memory 2.0 GB"
photon flavor create --name "small-disk"
--kind "ephemeral-disk" --cost "ephemeral-disk 1.0 COUNT"
```

6.2.6 Create a Subnet for the Cluster

Photon Platform combines Photon Controller with NSX to let you create infrastructure as a service—in this case, a virtual subnet for a Kubernetes cluster.

Before creating the subnet, however, you will need to identify an NSX Tier-1 router for the subnet by listing the available routers:

```
photon router list
```

From the list, copy the ID of the router that you want to use for the subnet and add it as the argument of the router option (`-r`) in the following command, which creates the subnet for the cluster:

```
photon subnet create -n myTestSubnet -d description -i 192.168.192.0/24 -r <router-ID>
```

Example:

```
photon subnet create -n myTestSubnet -d description
-i 192.168.192.0/24 -r 28db58b854cecc8dbb4e1
```

Finally, list the subnets so that you can obtain the ID of the subnet you just created for the cluster. You will need the subnet's ID when you create the cluster in the next step:

```
photon subnet list
```

6.2.7 Creating the Kubernetes Cluster

You are now ready to create a Kubernetes cluster by running the `photon service create` command. First, however, take a moment to look at the command's help and compare the differences in required options between clusters that use NSX networking and those that do not.

```
photon service create --help
```

When you create a Kubernetes cluster that uses NSX virtual networking, for example, you do not specify the IP addresses for the Kubernetes master, the load balancer, the etcd nodes, DNS, the gateway, or the netmask. The master node, load balancer, and etcd nodes automatically receive floating IP addresses. After the cluster is deployed, you can run the `photon service show` command to obtain information about the cluster, including its IP addresses.

Some information, however, must be supplied when you create the cluster:

- The number of masters, which must, at present, equal 1 or 2.
- The number of etcd nodes, which must equal either 1 or, for high availability, 3.

Here, then, is the command to create a Kubernetes cluster that uses NSX for networking. Note that, for the argument of the `w` option, the command uses the example ID of the subnet from earlier. Replace the example ID with your own.

```
photon service create --name k8-service --type KUBERNETES
--subnet_id 28db58b854ced1c9521e8 --worker_count 1
--number-of-etcds 3 --number-of-masters 1
--container-network 10.10.0.0/16
--master-vm-flavor "small-vm"
--worker-vm-flavor "small-vm"
--disk_flavor "small-disk"
```

Keep in mind that the flavors are not required but are added to constrain the size of the resources; if you do not specify the flavors, the defaults are used.

If the `service create` command prompts you for several inputs, you can press `Enter` to accept the defaults, or you can specify the values that you want. For example, you are prompted for the SSH key for the VMs. You can leave it empty or supply your own.

6.2.8 Getting Information About the New Cluster

You can get information about the cluster by running the following commands.

First, get the ID of the cluster:

```
photon service list
```

Next,run the following command, replacing the example ID with that of your newly created cluster:

```
photon service show 5b879631-3c28-4f54-af09-7cda100d9e42
Service ID:             5b879631-3c28-4f54-af09-7cda100d9e42
  Name:                 k8-service
  State:                READY
  Type:                 KUBERNETES
  Image ID:             de8e2b9d-0a82-4973-badc-3447fe4a4ed8
  Kubernetes IP:        172.22.0.11
  Worker count:         1
  Extended Properties:  map[master_ips:172.22.0.10
  etcd_ips:172.22.0.8
  service_ui_url:https://172.22.0.11:6443/ui
  pc_external_uri:https://172.20.0.55 client_linux_386_
...
```

The `photon service show` command's output contains key information:

- The Kubernetes IP address is the IP address that is assigned to the load balancer—the IP address that you connect to manage the cluster.
- The IP address to connect to the Kubernetes Dashboard.

6.2.9 Creating the Cluster with an API Call

Instead of using the `photon service create` command to set up a cluster, you can use the Photon Controller API. The following payload of an API call, for example, establishes an NSX-powered Kubernetes cluster. An

API call assumes that you have obtained an access token to work with the UI; see Using the API.

```
{
  "name": "test",
  "workerCount": 1,
  "type": "KUBERNETES",
  "extendedProperties": {
    "number_of_masters": 1,
    "number_of_etcds": 1,
    "container_network": "10.0.2.0/16"
  },
  "subnetId": "28db58b854ced1c9521e8",
  "workerBatchExpansionSize": 1
}
```

6.2.10 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:

`https://<ip-address-of-photon-load-balancer>:4343`

In the web interface, locate your tenant and the project that contains the NSX-powered Kubernetes cluster. Once you've reached the project, you can click `Clusters` and then click the name of a cluster to see its settings.

6.2.11 Opening the Kubernetes Dashboard

Now you're ready to open the Kubernetes Dashboard to work with your cluster. For instructions on how to obtain authentication information to open the dashboard, see Opening the Kubernetes Dashboard.

After you connect to the dashboard, you can run an application in your Kubernetes cluster; see Deploying an nginx Web Server on a Kubernetes Cluster.

6.3 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.

6.3.1 Downloading the Base Image for a Cluster

You can download the base image from the following location:

- Kubernetes

6.3.2 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.

6.3.3 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.

6.4 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.

6.4.1 Spinning Up a Cluster for Tomcat

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

```
photon service 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
--load-balancer-ip 192.168.209.34
--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.

6.4.2 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. To download the `kubectl` utility, see Installing and Setting Up kubectl.

6.4.2.1 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.

6.4.2.2 Check Out the Application

We can view our Tomcat application at the following URL:

```
http://192.168.209.35:30001
```

6.4.3 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
```

6.4.4 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 service delete <cluster_uuid>
```

6.5 Managing Clusters

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

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

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

```
photon service --help
```

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

```
photon service show -h
```

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

```
photon service --help
NAME:
   photon service - Options for services
USAGE:
   photon service command [command options] [arguments...]
COMMANDS:
     create               Create a new service
     show                 Show information about a service.
     list                 List services
     list-vms             List the VMs associated with a service
     resize               Resize a service
     delete               Delete a service
     trigger-maintenance  Start a background process to recreate
                          failed VMs in a service
     cert-to-file         Save the CA Certificate to a file with
                          the specified path if the certificate exists
     change-version       Configure the service to use specified image id
     get-kubectl-auth     Generate the kubectl command for authentication
OPTIONS:
   --help, -h  show help
```

6.6 Using Harbor with Kubernetes

Here's how to set up VMware Harbor to work with a Kubernetes cluster running on Photon Platform. Harbor is a local registry that stores, protects, and distributes Docker images.

Harbor includes such enterprise features as a web administrative interface, role-based access control, image replication, and Lightwave authentication.

On Photon Platform, Harbor can be deployed on Kubernetes. First, deploy a Harbor cluster. The admin password must contain at least an uppercase letter, a lowercase letter, and a digit.

```
photon service create --type HARBOR
       --master-ip <HARBOR-STATIC-IP>
       --dns <DNS-IP> --gateway <GATEWAY-IP> --netmask <NETMASK>
       --ssh-key ~/.ssh/id_rsa.pub --admin-password password
```

Here's an example:

```
photon service create --type HARBOR
     --master-ip 198.51.100.22
     --dns 192.0.2.1 --gateway 192.0.2.77 --netmask 255.255.0.0
     --ssh-key ~/.ssh/id_rsa.pub --admin-password Secret1!
```

Second, find the self-signed CA certificate used by Harbor. The `photon service cert-to-file` command can loate the self-signed CA certificate. Here's the output of the command's help:

```
photon service cert-to-file --help
NAME:
photon service cert-to-file - Save the CA Certificate to a file
with the specified path if the certificate exists
USAGE:
photon service cert-to-file cluster-id file_path
DESCRIPTION:
If a cluster has a CA certificate, this extracts it and saves
```

```
it to a file. If the specified file path doesn't exist, it will create
a new file with the specified pathThis is useful when using using Harbor,
which uses a self-signed CA certificate. You can extract the CA certificate
with this command, and use it as input when creating a Kubernetes cluster.
```

And here's an example:

```
photon service cert-to-file 9c246b84-8362-37e4-af58-35da7332e926 /home/photon/harbor.cert
```

Third, create a Kubernetes cluster that refers to the Harbor certificate with the `registry-ca-cert` option:

```
photon service create --namen my-cluster --type KUBERNETES
       --master-ip <HARBOR-STATIC-IP>
       --etcd1 <ETCD-STATIC-IP>
       --worker_count 5
       --vm_flavor cluster-master-vm
       --dns <DNS-IP> --gateway <GATEWAY-IP> --netmask <NETMASK>
       --container-network 10.2.0.0/16
       --ssh-key ~/.ssh/id_rsa.pub
       --registry-ca-cert /home/photon/harbor.cert
```

Here's an example:

```
photon service create --name kube-cluster1 --type KUBERNETES
     --master-ip 198.51.100.22
     --etcd1 198.51.100.75
     --worker_count 5
     --vm_flavor cluster-master-vm
     --dns 192.0.2.1 --gateway 192.0.2.77 --netmask 255.255.0.0
     --container-network 10.2.0.0/16
     --ssh-key ~/.ssh/id_rsa.pub
     --registry-ca-cert /home/photon/harbor.cert
```

Finally, you should set up Docker to use the Harbor registry securely. To pull and push images from your workstation or virtual machine, you must first configure Docker. Here's an example of how to do so on a Linux VM running the Docker daemon:

- Make sure the option "-insecure-registry" is not present.
- Copy the ca.crt file from Harbor (see above) to /etc/docker/certs.d/, where is the IP address of your Harbor registry.

Then log into Docker:

```
docker login -u admin -p <admin_passowrd> http://203.0.113.20
```

Tag and push your images:

```
docker tag frontend 203.0.113.56/library/frontend
docker push 203.0.113.56/library/frontend
```

For instructions on how to set up Docker on a Mac, see the Docker documentation.

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:

```
photon -h
NAME:
   photon - Command line interface for Photon Controller
USAGE:
   photon [global options] command [command options] [arguments...]
VERSION:
   Git commit hash: 11f5e4c
COMMANDS:
     auth              options for auth
     system            options for system operations
     target            options for target
     tenant            options for tenant
     host              options for host
     datastore         options for datastore
     deployment        options for deployment
     image             options for image
     task              options for task
     flavor            options for flavor
     project           options for project
     disk              Options for persistent disks
     vm                options for vm
     service, cluster  Options for services
     router            options for router
     subnet            options for subnet
     zone              options for 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 value, -l value  write logging information into a logfile at
                               the specified path
   --output value, -o value    select output format
   --detail, -d                print the current target, user, tenant, 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.

For instructions, see the Virtual SAN with Photon Controller Setup Guide.

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:

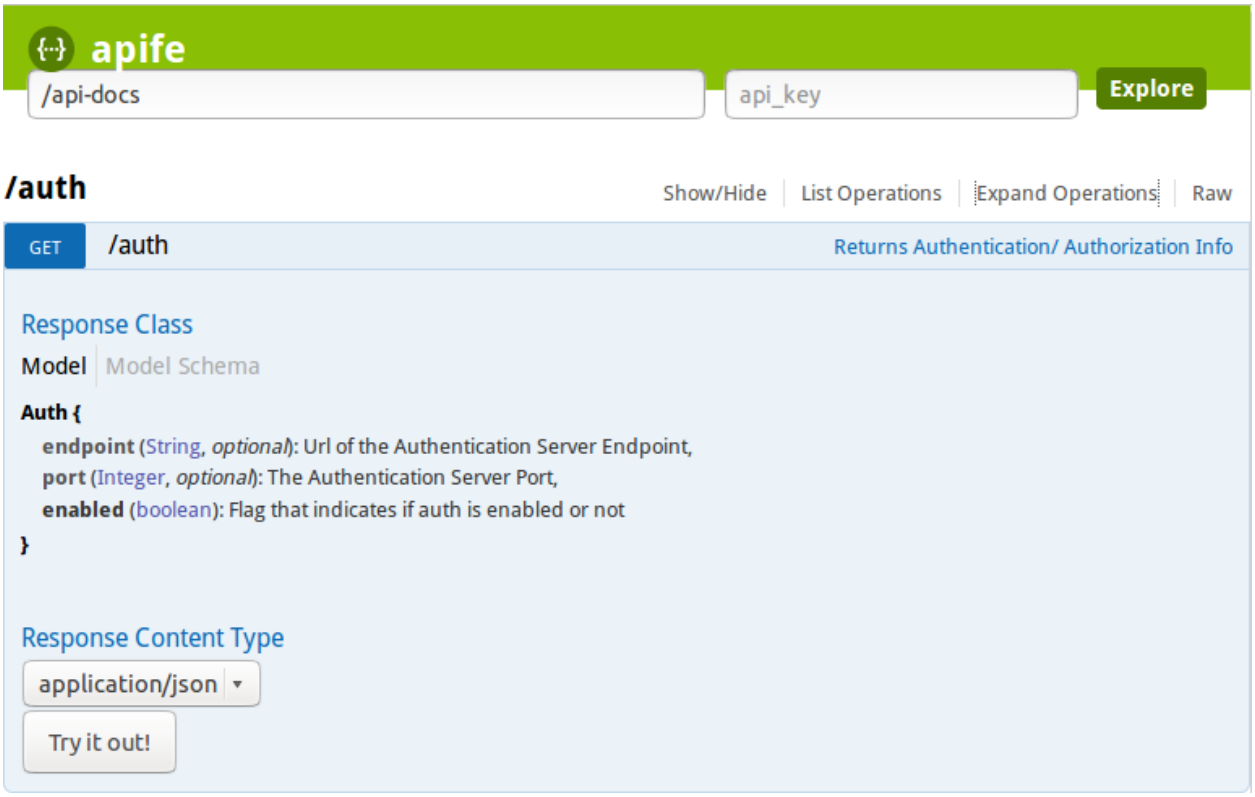

Figure 13: alt text

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 examples 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. `/v1/system/auth`: This allows you to get information about how to contact Lightwave to get tokens.
2. `/v1/system/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 passsword '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/v1/system/auth | jq .
{
   "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_photon at_groups". Note that this is subject to change, and `photon` might be a different name in your release.

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_photon 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@example.com&password=Passw0rd!&scope=openid
  offline_access rs_photon 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": "PHOTON-CONTROLLER-CORE",
      "status": "READY",
    ... output trimmed ...
  "status": "READY"
}
```

10 Troubleshooting and Maintenance

10.1 Ports and Protocols

Photon Platform uses the following ports and protocols. To view the role the ports and protocols play in Photon Platform, see the Photon Platform Transport Layer diagram.

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

Port 4343: Photon Controller uses Port 4343 for HTTPS connections to the management web interface or its load balancer when Photon Controller is set up to use Lightwave authentication.

Port 9000: Photon Controller uses Port 9000 for unauthenticated connections to the RESTful API for back-end operations if Lightwave is not installed.

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

Port 22: System administrators can connect to virtual machines in ESXi hypervisors with SSH over TCP on Port 22.

Port 20000: Photon Platform's management service uses TCP over Port 20000 for its back-end HTTP communication.

Port 20001: Photon Platform's management service uses TCP over Port 20001 for its back-end HTTPS communication.

Port 2015: Photon Controller uses TCP over Port 2015 for the VMware DNS service.

Port 19000: Photon Controller uses TCP over Port 19000 for communications among VMs in the Photon Controller management plane for such services Xenon, the cloud store, and the housekeeper.

Port 8835: The Photon Controller management plane uses TCP over Port 8835 for its host client to communicate with vSphere hypervisors.

10.1.1 External Systems

Port 53 for DNS: Photon Platform uses TCP or UDP over Port 53 to communicate with an external name server for DNS.

Port 123 for NTP: Photon Platform uses UDP over Port 123 to communicate with an external NTP server.

Port 514 for syslog: Photon Platform's syslog service uses TCP and UDP over Port 514 to communicate with an external support system, such as VMware vRealize LogInsight.

10.1.2 Security Systems

Port 443: Photon Controller uses TCP over Port 443 to authenticate users with Lightwave.

Port 389: Photon Controller uses TCP over Port 389 to connect with the Lightwave LDAP service.

Port 636: Photon Controller uses TCP over Port 636 to connect with the Lightwave LDAPS service.

Port 88: Photon Controller uses TCP and UDP over Port 88 to connect with the Kerberos security protocol to the Lightwave key distribution center.

Port 2020: Photon Controller uses TCP over Port 2020 as part of the authentication framework.

Port 445: Lightwave uses UDP over Port 445 to communicate with the Microsoft directory service when Lightwave is integrated with Microsoft Active Directory. In such a case, Active Directory is an external system.

Port 2012: Lightwave uses TCP over Port 2012 for its directory service.

Port 2014: Lightwave uses TCP over Port 2014 for its certificate service.

Port 53: Lightwave uses TCP and UDP over Port 53 for DNS.

10.1.3 Shared Storage

Port 2049 for NFS: The ESXi hosts in a Photon Platform cluster use TCP and UDP over Port 2049 to connect with an NFS storage system.

Port 111 for rpcbind: For NFS, an ESXi host uses Port 111 over TCP and UDP for rpcbind.

Port 3260 for iSCSI: The ESXi hosts in a Photon Platform cluster use TCP over Port 3260 to connect with an iSCSI storage system.

10.1.4 VMware vSAN Storage

Port 2233: The ESXi hosts in a Photon Platform cluster use Port 2233 for transporting data to vSAN.

Ports 12345 and 23451: The ESXi hosts in a Photon Platform cluster use Ports 12345 and 23451 over UDP for clustering with vSAN.

10.1.5 NSX

Port 443: Network administrators connect to the NSX Manager with TCP on Port 443 for HTTPS connections to the web interface. The NSX-T API also uses TCP on Port 443 for communication with Photon Controller's core services.

Port 5672: The NSX-T Manager uses TCP over Port 5672 to communicate with ESXi hypervisors and the NSX Controllers.

Port 50 and 500: The NSX Controllers use IP and UDP over Ports 50 and 500 for the IPSec service.

Port 1234 The NSX Controller Cluster uses TCP over Port 1234 for the Icp-plane client's communication with the vSphere hypervisors.

10.2 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.

10.2.1 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.

10.2.2 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
```

10.3 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.

10.3.1 Log Files

The log files for the deployment reside in the following location on the installer VM. Recall from the installation guide that the default password for the root account is `changeme` if you haven't changed it already.

- /var/log/photon-installer.log

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

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

ESXi errors related to the agent's calls are stored in the following location on each target ESXi host:

- /scratch/log/syslog.log

The log files on the Photon Controller management VMs might contain useful troubleshooting information about API calls that fail, such as the call to add an ESXi host to the cluster. The management VM's log resides at this location:

- /var/log/photon-platform/photon-controller.log

You can access the log file through the VM's console in the vSphere Web Client. To connect to the management VM with SSH to view the log, you must first connect to the console and change the SSH configuration to permit root login; see Permitting Root Login with SSH.

10.3.2 Image Upload Failure

Verify the file path you entered.

10.3.3 Reserve_Resorce Error

A Reserve_Resource error may look something like this:

```
2015/11/10 01:08:12 photon: Task '48b474d6-1773-4dca-a5cb-7a45a38e1904' is in error
state. Examine task for full details.API Errors: [photon: { HTTP status: '0', code:
'InternalError', 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 "InternalError",
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 no 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

10.3.4 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 ESXi host by using the vSphere Web Client.

11 Appendix I: Command-Line Examples

11.0.5 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
   datastore    options for datastore
   deployment   options for deployment
   image        options for image
   task         options for task
   flavor       options for flavor
   project      options for project
   disk         options for disk
   vm           options for vm
   service      Options for services
   router       options for router
   subnet       options for subnet
   zone         options for 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 value, -l value  write logging information into a logfile
                               at the specified path
```

```
   --output value, -o value    select output format
   --detail, -d                print the current target, user, tenant and project
   --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
   iam                  options for identity and access management
   quota                options for tenant quota
   help, h              Shows a list of commands or help for one command

OPTIONS:
   --help, -h show help
```

11.0.6 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.

11.0.7 IDs

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

11.0.8 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 -c 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'
```

11.0.9 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
```

11.0.10 Setting a 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
```

11.0.11 Quota

You assign quota to control the resource allocations granted to tenants and projects.

A quota must specify the number of VMs that can be created as well as the total amount of RAM consumed by those VMs. If a flavor is used for VM creation, all of its costs must be in the quota including ephemeral and persistent disk capacities. It's possible to have user-defined quota 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 tenant with quota assigned (see above, or use the –tenant flag):

Usage `photon tenant create cloud-dev --limits "<LIMITS>"`

Example:

```
% photon -n tenant create --name cloud-dev
         --limits "vm.memory 2000 GB, vm 1000 COUNT, ephemeral-disk.capacity 2000 GB"
32ad527e-d21a-4b2a-a235-b0883bd64354
```

Creating a tenant with user-defined resources:

```
% photon -n tenant create --name cloud-dev
         --limits "vm.memory 2000 GB, vm 1000 COUNT,
         ephemeral-disk.capacity 2000 GB, vm.potrzebie 250 COUNT"
32ad527e-d21a-4b2a-a235-b0883bd64354
```

11.0.12 Projects

A project is owned by a tenant and all VMs are created within a project. Each project is assigned with a quota that controls the total resources that can be used. See above for more information about quota.

A project has a set of limits. These are specified just like the quota above, but they must not exceed the limits in the tenant.

Creating a project:

Usage: `photon project create <PROJECT-NAME> --limits <LIMITS>`

```
% photon -n project create 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
```

11.0.13 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-88ce-4f09-9b5d-49662d21e56c  cloud-disk
ephemeral-disk  ephemeral-disk:1:COUNT
ddfb5be0-3355-46d3-9f2f-e28750eb201b  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
```

11.0.14 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
ID     Name       State  Size(Byte)   Replication_type  ReplicationProgress  SeedingProgress
8d0... 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 8d0b9...
```

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:
```

11.0.15 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://198.0.2.41:9000'
VM ID:  86911d88-a037-4576-9649-4df579abb88c
  Name:          vm-1
  State:         STARTED
  Flavor:        cloud-vm-small
  Source Image: 8d0b9383-ff64-4112-85db-e8111e2269fc
  Host:          10.160.98.190
  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
fabb923...  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
```

11.0.16 Hosts

Adding an ESX host:

Usage: `photon host create -u <USER-NAME> -p <PASSWORD>  -i <ADDRESS> --tag <CLOUD|MGMT> -d <DEPLOYMENT-ID>`

```
% photon -n host create -u root -p MY-PASSWORD
         -i 198.51.100.41 --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

Here's a complete example YAML file that installs Photon Platform on a VMware NSX-T network. The installation includes Lightwave, a load balancer, and VMware vSAN.

Additional templates and explanations of the fields appear in the Photon Controller Quick Start Guide.

```
compute:
  hypervisors:
    esxi-1:
      hostname: "pc-1"
      ipaddress: "198.51.100.1"
      dns: "198.51.100.12"
      credential:
        username: "root"
        password: "Secret1!"
    esxi-2:
      hostname: "pc-2"
```

```
      ipaddress: "198.51.100.12"
      dns: "198.51.100.12"
      credential:
        username: "root"
        password: "Secret1!"
    esxi-3:
      hostname: "pc-3"
      ipaddress: "198.51.100.3"
      dns: "198.51.100.12"
      credential:
        username: "root"
        password: "Secret1!"
    esxi-4:
      hostname: "pc-4"
      ipaddress: "198.51.100.4"
      dns: "198.51.100.12"
      credential:
        username: "root"
        password: "Secret1!"
    esxi-5:
      hostname: "pc-5"
      ipaddress: "198.51.100.8"
      dns: "198.51.100.12"
      credential:
        username: "root"
        password: "Secret1!"
lightwave:
  domain: "example.com"
  credential:
    username: "administrator"
    password: "Secret123$"
  controllers:
    lightwave-1:
      site: "new york"
      appliance:
        hostref: "esxi-1"
        datastore: "datastore1"
        memoryMb: 2048
        cpus: 2
        credential:
          username: "root"
          password: "Secret1!"
        network-config:
          type: "static"
          hostname: "lightwave-1.example.com"
          ipaddress: "198.51.100.12"
          network: "NAT=VM Network"
          dns: "198.51.100.12,198.51.100.13"
          ntp: "203.0.113.1"
          netmask: "255.255.252.0"
          gateway: "198.51.100.253"
    lightwave-2:
      site: "cambridge"
      partner: "198.51.100.13"
```

```
      appliance:
        hostref: "esxi-1"
        datastore: "datastore1"
        memoryMb: 2048
        cpus: 2
        credential:
          username: "root"
          password: "Secret1!"
        network-config:
          type: "static"
          hostname: "lightwave-2.example.com"
          ipaddress: "198.51.100.13"
          network: "NAT=VM Network"
          dns: "198.51.100.12,198.51.100.13"
          ntp: "203.0.113.1"
          netmask: "255.255.252.0"
          gateway: "198.51.100.253"
photon:
  imagestore:
    img-store-1:
      datastore: "datastore1, datastore2"
      enableimagestoreforvms: "true"
  cloud:
    hostref-1: "esxi-5"
    hostref-2: "esxi-3"
  administrator-group: "example.com\Administrators"
  syslog:
    ipaddress: "198.51.100.23"
  controllers:
    pc-1:
      appliance:
        hostref: "esxi-1"
        datastore: "datastore1"
        memoryMb: 2048
        cpus: 2
        credential:
          username: "root"
          password: "Secret1!"
        network-config:
          type: "static"
          hostname: "pc-1.example.com"
          ipaddress: "198.51.100.14"
          network: "NAT=VM Network"
          netmask: "255.255.252.0"
          dns: "198.51.100.12,198.51.100.13"
          ntp: "203.0.113.1"
          gateway: "198.51.100.253"
    pc-2:
      appliance:
        hostref: "esxi-1"
        datastore: "datastore1"
        memoryMb: 2048
        cpus: 2
        credential:
```

```
          username: "root"
          password: "Secret1!"
        network-config:
          type: "static"
          hostname: "pc-2.example.com"
          ipaddress: "198.51.100.15"
          network: "NAT=VM Network"
          netmask: "255.255.252.0"
          dns: "198.51.100.12,198.51.100.13"
          ntp: "203.0.113.1"
          gateway: "198.51.100.253"
vsan:
  vsan-1:
    appliance:
      hostref: "esxi-1"
      datastore: "datastore1"
      memoryMb: 2048
      cpus: 2
      credential:
        username: "root"
        password: "Secret1!"
      network-config:
        type: "static"
        hostname: "vsan-1.example.com"
        ipaddress: "198.51.100.20"
        network: "NAT=VM Network"
        netmask: "255.255.252.0"
        dns: "198.51.100.12,198.51.100.13"
        ntp: "203.0.113.1"
        gateway: "198.51.100.253"
loadBalancer:
  load-balancer-1:
    appliance:
      hostref: "esxi-1"
      datastore: "datastore1"
      memoryMb: 2048
      cpus: 2
      credential:
        username: "root"
        password: "Secret1!"
      network-config:
        type: "static"
        hostname: "lb-1.example.com"
        ipaddress: "198.51.100.21"
        network: "NAT=VM Network"
        netmask: "255.255.252.0"
        dns: "198.51.100.12,198.51.100.13"
        ntp: "203.0.113.1"
        gateway: "198.51.100.253"
nsxconfig:
  ipaddress: "203.0.0.1"
  credential:
    username: "root"
    password: "Secret1!"
```

```
privateIpRootCidr: "203.0.0.1"
floatingIpRootRange: "198.51.100.2-198.51.100.20"
t0RouterId: "123"
edgeCLusterId: "456"
overlayTransportZoneId: "123"
tunnelIpPoolId: "123"
hostUplinkPnic: "vmnic4"
hostUplinkVlanId: "0"
dnsServerAddresses: "198.51.100.12"
```