WHITE PAPER – **MAY 2017**

DEPLOYING CLOUD-NATIVE APPLICATIONS WITH PHOTON OS

Technical Overview

vmware®

Table of Contents

Introduction

Project Photon OS™ is an open-source minimalist Linux operating system from VMware optimized for cloud computing platforms, VMware vSphere® deployments, and applications native to the cloud.

This technical overview describes Photon OS, illustrates its role in VMware's cloud computing architecture, and explains the part it plays in solving a range of problems that would otherwise limit the deployment of cloud-native applications.

The problems that Photon OS addresses stem from the digital transformation reshaping business. Companies of all types are under pressure to create innovative software that engages their customers. The digital technologies at the source of this shift are cloud computing, mobile devices, and data analytics. Companies can exploit these technologies to lower costs, connect with customers, and improve their bottom line.

But reinventing a traditional company, or even a technology company, as a contemporary software-centric enterprise requires the creation of applications that run in the cloud and the infrastructure and tools to build them. To accelerate the development of innovative software and to adapt to changes in the marketplace, you are likely to need such technologies as containers, microservices, distributed systems, orchestration tools, and virtualization.

For their cloud infrastructure, technology-savvy companies seek robust, API-driven, open source solutions that scale to handle large volumes of data at pace. But putting in place scalable, flexible infrastructure that fosters the development and deployment of cloud applications can be complex, difficult, and costly.

The fast track to cost-effectively adopt containers is to transform your existing virtualized infrastructure into a flexible, scalable mode of deploying cloud-native applications. Photon OS, as a run-time environment optimized for containers, plays a central role in this approach, especially when it is combined with VMware vSphere or VMware Photon™ Platform. Designed with a focus on security and cloud computing, Photon OS helps overcome the problems that businesses face as they seek to implement cloud-native applications.

A Minimalist Container Host

Photon OS is a Linux container host optimized for vSphere, Amazon Elastic Compute Cloud, Google Compute Engine, and other cloud platforms. Photon OS is also the default container host on VMware Photon Platform—a multi-tenant, API-driven system for delivering Kubernetes and infrastructure as a service. A container host runtime environment provides just enough of a Linux operating system to efficiently run containers.

Photon OS combines its compactness with the extensibility and in-place updates of a mainstream Linux distribution. These hybrid characteristics give Photon OS the flexibility to adapt to changing requirements—especially those associated with Docker containers.

The Rise of Containers

A container is a process that runs on a host with its own isolated application, file system, and networking. Enterprises are increasingly moving toward container-based architectures to develop applications faster, increase automation, and reduce server costs.

Because each container is self-describing, specifying the computing and networking resources that it needs, a container packages an application in a reproducible way: It can be distributed and reused with minimal effort, yielding both economies of scale and efficiencies in reuse.

Photon OS includes the open source version of Docker to streamline the workflow of getting a container running in a hypervisor. A developer can install a hypervisor such as VMware Fusion® on a laptop, replicate a cluster of virtual machines, and then, with Photon OS, build containerized applications with microservices, an architectural pattern that isolates the code performing a function into an independently deployable service.

Moving Containerized Applications from Development to Production

Because Photon OS is optimized to work with VMware vSphere, VMware Workstation Pro™, and VMware Fusion, Photon OS empowers you to seamlessly migrate container-based applications from development to production while—unlike other systems—helping to maintain the isolation and security of the application running in the container.

Running applications in containers on Photon OS machines integrated with vSphere overcomes a significant problem that plagues containerized applications: They are difficult to deploy into production securely. The security of virtual machines running in vSphere coupled with the security-first design of Photon OS and your own network security measures helps establish production-level security for the containerized application.

There is another hurdle that keeps developers from deploying containers in production: IT operations. At many companies, developers and IT are separate entities. Without the cooperation and commitment of IT, developers can do little to move their Docker workloads into production. By tying in seamlessly with VMware vSphere, which often forms the basis of the production environment that is owned and operated by IT, Photon OS paves the way to put containerized applications into production.

Photon OS takes this connection with virtual infrastructure one step further—and helps alleviate IT concerns over security—by integrating with VMware's Project Lightwave™, a security suite for cloud-native applications.

The minimalist architecture of Photon OS further segregates and streamlines the process of moving containers from development to production.

The Minimalist Architecture of Photon OS

In the context of containerized applications, Photon OS embodies the design principles of minimalism by coming in two versions: a minimal version and a full version. Each version contains only the elements necessary to fulfill its use case for containers.

The minimal version of Photon OS is a lightweight container runtime best suited to managing and hosting containers. The minimal version contains just enough functionality to manage containers while remaining a fast runtime environment. The minimal version is primed to work with appliances, and it comes in the OVA format for rapid deployment on hypervisors. The installation of the minimal version stands at about 380 MB on disk.

The full version of Photon OS includes additional packages to help create containerized applications. For merely running containers, the full version is excessive. It is targeted at helping you develop, test, and package an application that runs in a container or in the cloud.

Both versions of Photon OS yield several benefits:

- An improvement in resource-efficiency by using smaller server builds
- A reduction in security risks by removing vulnerable components
- A decrease in management effort by having fewer components to update

In the vein of minimalism, two more characteristics of Photon OS stand out: It manages services with systemd, and it manages packages with a compact, yum-compatible package manager called tdnf, for Tiny Dandified Yum, a reference to the dandified yum component slated to be the new update manager for Fedora.

Systemd

By using systemd, Photon OS adopts a contemporary Linux standard to manage system services. Photon OS bootstraps the user space and concurrently starts services with systemd—an architecture that differs from traditional Linux systems.

Systemd uses a dependency tree of targets to determine which services to start when. Combined with the declarative nature of systemd commands, systemd targets reduce the amount of code needed to run a command, leaving you with code and scripts that are easier to maintain and faster to execute.

Package Management with Tiny Dandified Yum

On Photon OS, tdnf is the default package manager. Tdnf keeps the operating system as small as possible while preserving yum's robust package-management capabilities. The TDNF component can help identify the packages that a file or executable depends on. Here's an example:

```
tdnf provides /usr/bin/docker
docker-1.13.1-1.ph1.x86_64 : Docker
Repo: photon-updates
```

With tdnf, you can remove unnecessary libraries, files, and executables if your application does not need them, producing your own just-enough operating system, or JeOS. For instructions on how to use tdnf, see the Photon OS Administration Guide.

Tdnf is a compact open source C implementation of DNF package manager that is made available on GitHub by VMware. The size of the tdnf installation comes in under 250 KB, which helps keep the on-disk size of Photon OS small. In contrast, yum, which relies on Python, would add an additional 30 MB of packages. Tdnf is one of Photon's underlying system tools that simplify life-cycle management.

Life-Cycle Management

Several design elements of Photon OS, including tdnf, play a role in its strong life-cycle management features. Overall, Photon OS seeks to reduce the burden and complexity of managing clusters of Linux machines by including an efficient packaging model, extensibility, and centralized administration in its fundamental design. Here are some of Photon's design elements that simplify life-cycle management:

- Atomic updates with RPM-OSTree
- Incremental stateful updates (RPMs)
- Package repositories curated by VMware
- Extensible distribution: You can add and remove functionality incrementally
- Signed YUM repositories

These design elements come together to make it easy to update the system, perform in-place upgrades, and refresh installed packages like Docker and Kubernetes. Because the minimal version of Photon OS is designed to be mainly a read-only OS, it can be replaced in an atomic fashion.

Centralized Linux Management with RPM-OSTree

RPM-OSTree is a package-aware file tree replication system that keeps Linux machines synchronized with the latest bits in a predictable and reliable way. To maintain consistency across file systems, RPM-OSTree uses a git-like repository that records the changes to any file and replicates them to any subscriber.

RPM-OSTree lets you compose packages and other configuration options into a file tree on a server for atomic updates. The hosts download the file tree from the server and incrementally upgrade when the file tree changes. In this way, RPM-OSTree delivers identical, predictable installed systems to solve the problems that commonly plague system administrators as they struggle to maintain a farm of computers with different packages, files, and configurations.

When you install Photon OS from its ISO, it offers two installation options to take advantage of OSTree, a server and a host. The OSTree Host installation option creates a Photon OS instance that obtains its packages from an RPM-OSTree server. The host instance's packages and library are then centrally managed by the server.

The OSTree Server installation option creates an instance of a server that manages the file system tree of the OSTree hosts. Creating a Photon OSTree Server establishes a new repository and management node for the OSTree hosts. The OS OSTree Server then manages the hosts as versioned, atomic entities to simplify life-cycle management and security on an enterprise scale. For more information, see RPM-OSTree in the Photon OS wiki.

Curated Package Repositories

VMware curates the packages in the Photon OS repositories. The packages of key clustering technologies—Kubernetes, Docker, Swarm—are updated with the latest versions on a regular basis. Patches for key applications that require heightened security, such as OpenSSL, are added to the updates repository as soon as an upstream patch is available and verified.

The curated packages yield two results: An operating system primed with the latest open source projects for building cloud-native applications, and a suite of common components maintained to a secure standard.

The open source packages in Photon OS for building cloud-native applications provide the breadth and flexibility to address a range of use cases, including innovative use cases on the cutting edge of containerized applications.

The default installation of Photon OS includes four yum-compatible repositories plus the repository on the Photon OS ISO when it's available in a CD-ROM drive:

```
ls /etc/yum.repos.d/
lightwave.repo
photon-iso.repo
photon-updates.repo
photon.repo
```

The Photon ISO repository (photon-iso.repo) contains the installation packages for Photon OS. All the packages that Photon builds and publishes reside in the RPMs directory of the ISO when it is mounted. The RPMs directory contains metadata that lets it act as a yum repository. Mounting the ISO gives you all the packages corresponding to a Photon OS build. If, however, you built Photon OS yourself, the packages correspond only to your build, which will typically be the latest packages if you built it from the source code. In contrast, the ISO that you obtain from the Bintray web site contains only the packages that are in the ISO at the point of publication. As a result, the packages may no longer match those on Bintray, which are updated regularly.

The updates repository (photon-updates.repo) is irrelevant to a major release until after the release is installed. Thereafter, the updates repository holds the updated packages for that release. The repository, that is, points to updates for the installed version, such as a version of Kubernetes that supersedes the version installed during the major release.

The main Photon OS repository (photon.repo) contains all the packages that are built from the ISO or from another source. This repository points to a static batch of packages and spec files at the point of a release. This repository includes an open source authentication engine called Likewise Open as well as other free VMware software.

The Lightwave repository (lightwave.repo) contains the packages that make up the VMware Lightwave security suite for cloud applications, including tools for identity management, access control, and certificate management. Lightwave is discussed in a later section.

You can set up your own repositories, either on a local server or in the cloud, to distribute applications. See the Photon OS Administration Guide.

Extensibility and Incremental Stateful RPM Updates

Photon OS performs incremental stateful updates of RPMs. As a DevOps manager, you can update an application such as Docker individually without having to update an entire branch, as some other operating systems require you to do.

The RPM-based approach to managing the operating system makes Photon OS extensible. As an extensible RPM-based distribution, you can add or remove applications individually. For example, you can take the minimal version and then just install postgres if that is all the support your application needs.

Security-Hardened Linux

The design of Photon OS prioritizes security. Photon OS secures itself with its build process, compiler settings, and PGP-signed packages and repositories. As a system administrator or DevOps manager, you can enforce security with vulnerability scans, the pluggable authentication modules, the Linux auditing service, and many other measures.

Photon OS also works with VMware's open source Lightwave project to set up a certificate store and a secure LDAP directory service for cloud-native applications. Another open source application, Likewise Open, integrates Photon OS machines with Microsoft Active Directory for authentication and access control.

Improving Security and Eliminating Vulnerabilities with Packages

As a streamlined Linux operating system that the Photon OS team compiles from source, Photon OS is hardened in part by its build process. The Photon OS team can audit packages, such as OpenSSL, to identify vulnerabilities before releasing the system. Vulnerabilities can be fixed by applying and testing security patches as soon as they become available.

Compiler Settings for Security

Photon OS incorporates key compiler settings that improve the security of userspace executables and libraries. The following compiler settings apply to nearly all packages in Photon OS.

Strong protection against buffer overflow attacks. Photon OS builds all executables with the -fpie and -fPIE flags to generate position independent code. In these flags, "pie" stands for position independent executable.

When Photon OS executes the ld command to load a program, Photon OS also uses the pie option to set the code as a position independent variable.

Photon OS uses address space layout randomization (ASLR) in the kernel to randomize the userspace address layout, a security technique that helps protect against buffer overflow attacks. ASLR prevents an attacker from going to an exploitable function in memory by randomizing the address space position of an executable. Placing the body of an executable in a random address in memory helps prevent return-to-libc-type attacks on a program, especially programs with static libraries.

Strong stack-smashing protection in the GNU Compiler Collection (GCC). Stack-smashing protection inserts buffers after local pointers and function arguments in the stack frame to reduce the corruption of pointers. Because a corrupt pointer can provide access to arbitrary memory locations, reducing the corruption of pointers improves security by helping to prevent access to arbitrary memory locations.

Photon OS implements the -fstack-protector-strong flag to balance security and performance. Although the -fstack-protector-strong flag protects more kinds of vulnerable functions than -fstack-protector, it does not protect every function, yielding better performance than -fstack-protector-all.

Extensive checking for buffer overflows in GCC functions that perform operations on memory and strings. By implementing the -D_FORTIFY_SOURCE=2 compiler flag for the GNU Compiler Collection (GCC), Photon OS protects C and C++ code when it is compiled and when it is run.

The -D_FORTIFY_SOURCE=2 flag imposes best practices for libc functions by checking buffer lengths and memory regions when code is compiled or run. Part of the power of the -D_FORTIFY_SOURCE=2 flag stems from its ability to improve security by preventing common buffer overflows with no or non-measurable runtime overhead. The -D_FORTIFY_SOURCE=2 features improves security because an undetected buffer overflow could, for instance, give an attacker the opportunity to exploit a flaw in a program while copying a string or performing some other function.

Preventing overwrite attacks with -z relro. Photon OS hardens the process of building and loading a program by implementing the -z relro parameter for the ld command. This option changes several ELF memory sections to read-only before turning control over to the program, helping to thwart some types of Global Offset Table and ".dtors" overwrite attacks. The Executable and Linkable Format (ELF) is a standard format for executables, object code, shared libraries, and core dumps.

Resolving all dynamic symbols with -z now before marking them read-only. Photon OS hardens the process of loading a program by implementing the -z now parameter for the ld command to resolve dynamic symbols so that the complete Global Offset Table can be marked read-only by the -z relro option. The Global Offset Table, or GOT, holds absolute addresses in private data to protect the position-independence and shareability of a program's code.

Hiding kernel pointers with kptr_restrict. Photon OS uses the kptr_restrict setting to place restrictions on the kernel addresses exposed through /proc and other interfaces. This setting hides exposed kernel pointers to prevent attackers from exploiting kernel write vulnerabilities.

Signed Packages and Repositories

Photon OS signs its packages and repositories with GPG signatures to bolster security. The GPG signature uses keyed-hash authentication method codes, typically the SHA1 algorithm and an MD5 checksum, to simultaneously verify the integrity and authentication of a package. A keyed-hash message authentication code combines a cryptographic hash function with a secret cryptographic key.

In Photon OS, GPG signature verification automatically takes place when you install or update a package with the default package manager, tdnf. If you have one of the RPMs from Photon OS, you can check the status of the SHA and MD5 for the package to verify that it has not been tampered with:

```
rpm -K /home/photon/stage/SRPMS/kubernetes-1.6...src.rpm
/home/photon/stage/SRPMS/kubernetes-1.6...src.rpm: sha1 md5 OK
```

And then you can view the SHA1 digest and the MD5 digest like this:

```
rpm -Kv /home/steve/workspace/photon/stage/SRPMS/kubernetes-1.6...src.rpm
/home/steve/workspace/photon/stage/SRPMS/kubernetes-1.6...src.rpm:
Header SHA1 digest: OK (89b55443d4c9f67a61ae0c1ec9bf4ece2d6aa32b)
MD5 digest: OK (51eee659a8730e25fd2a52aff9a6c2c2)
```

These truncated examples show that the Kubernetes package has not been tampered with.

Identity and Certificate Management with Lightwave Security Services

Photon OS integrates with Lightwave to provide security services. An open source project published on GitHub, Lightwave furnishes a directory service, a certificate authority, a certificate store, and an authentication service. You can use Lightwave to, for instance, join a Photon OS virtual machine to the Lightwave directory service and then authenticate users with the Kerberos security protocol.

On Photon OS, obtaining Lightwave is easy because Photon OS includes the Lightwave repository by default:

```
tdnf repolist

    repo id           repo name                          status
    photon-extras     VMware Photon Extras 1.0(x86_64)   enabled
    lightwave         VMware Lightwave 1.2(x86_64)       enabled
    photon-updates    VMware Photon Linux 1.0(x86_64)    Updates enabled
```

Project Lightwave comprises the following key elements to integrate security with a cluster of Photon OS machines:

- Directory Service. This standards-based, multi-tenant, multi-master, highly scalable LDAP v3 directory service enables an enterprise's infrastructure to be used by the most-demanding applications as well as by multiple teams.
- Certificate Authority. This directory-integrated certificate authority helps simplify certificate operations and key management across the infrastructure.
- Certificate Store. This endpoint certificate store holds certificate credentials.
- Authentication. This cloud authentication service supports Kerberos, OAuth 2.0/OpenID Connect (OIDC), WebSSO, SAML, and WSTrust for interoperability with other standards-based technologies in the data center.

These Lightwave modules can be used to help Photon OS machines comply with such regulations as FISMA, HIPAA, and PCI DSS.

Authentication and Access Control with Likewise Open

Lightwave includes Likewise Open, an open source agent that can connect Photon OS to Active Directory and authenticate users with their domain credentials.

Likewise Open joins Photon OS machines to an Active Directory domain to extend AD's identity management and authentication capabilities to Linux computers. When a user in Active Directory logs in to a Linux machine joined to Active Directory, Likewise Open authenticates the user's identity by using the highly secure Kerberos protocol, which can help you comply with federal and industry regulations for authentication and access control.

Since the Likewise Open package resides in the Lightwave repository included with Photon OS, installing Likewise Open is simple:

```
tdnf install likewise-open
Installing:
likewise-open         x86_64         6.2.9-0         12.33 M
Total installed size: 12.33 M
```

See the Likewise Open Github site.

Scanning for Vulnerabilities with cve-check

The full version of Photon OS includes an open source tool from Intel that checks for common vulnerabilities and exposures. The cve-check-tool scans distribution source packages for vulnerabilities listed in a locally cached copy of the National Vulnerability Database. In the minimal version of Photon OS, you can install the tool by using the package manager. For more information on the tool, see the cve-check-tool site on GitHub.

Securing User Sessions with PAM

Photon OS supports the Pluggable Authentication Modules, or PAM, to verify user identities and to manage the security of their login sessions. More specifically, Photon OS, like other contemporary Linux systems, includes modules to manage authentication, user accounts, user sessions, and user passwords. On Photon OS, PAM can control the complexity of user passwords, deny access to users during certain times, control access to the system, modify the access of group accounts, and impose other security measures on logins and user sessions.

Root Password Rules

Photon OS requires the root password to be set to a complex string containing no common words or names. Photon OS rejects a root password that contains simplistic patterns, common words, or words derived from the name of your account. The rules apply only to the root password, not other user and group accounts. For users and groups other than root, Photon OS can be set to require complex passwords through the pam_cracklib module.

Auditing

Because Photon OS emphasizes security, the Linux auditing service, auditd, is enabled and active by default on the full version of Photon OS. To help improve security, the auditd service can monitor file changes, system calls, executed commands, authentication events, and network access. You can, for instance, use the auditctl utility to set a rule that monitors the sudoers file for changes. After you implement an audit rule to monitor an event, the aureport tool generates reports to display information about the events. The reports can help demonstrate regulatory compliance.

Automating Deployment

Photon OS expedites large-scale deployments in virtualized environments. Photon OS works with the following technologies so you can deploy many Linux machines at once:

- OVA
- PXE Boot
- Kickstart
- PXE combined with Kickstart
- OSTree
- Vagrant

Rapid Deployment on VMware Platforms with OVA

The minimal version of Photon OS comes in an OVA version to expedite and automate its deployment in VMware ESXi™, VMware Workstation Pro, VMware Fusion, and VMware vSphere.

Photon OS is distributed as an Open Virtual Appliance (OVA) package, a TAR archive that contains an OVF package. The Open Virtualization Format (OVF) is a National ANSI standard for packaging software for virtual machines. The Photon OS OVA packages the operating system as a virtual appliance ready to be deployed as a virtual machine.

The VMware OVF Tool, a command-line tool to port OVF packages to VMware platforms, can automate the deployment of Photon OS VMs.

With the OVF Tool, you can extract the Photon OS OVA archive to an OVF package, deploy it to an ESXi host in vSphere, and power it on:

```
ovftool /ovfs/photon-custom-hw11-1.0-13c08b6-GA.ova /ovfs/photonos.ovf
ovftool --powerOn photonos.ovf vi://my.esx-machine.example.com/
```

Such commands could be combined into a script to automate the deployment of Photon OS machines in vSphere.

In addition to the speed with which you can deploy Photon OS in vSphere, the OVA version of Photon OS available on Bintray is optimized to run on VMware ESXi in vSphere. See the section on Photon OS performance on ESXi later in this paper.

PXE Boot

Photon OS works with the Preboot Execution Environment, or PXE, to boot by retrieving software from a PXE server over a network connection. PXE initiates the installation of the operating system from a distribution point without your intervention. For an example of how to deploy Photon OS by using PXE, see Network PXE Boot.

Kickstart

Photon OS supports kickstart for unattended installations through a CD-ROM or an HTTP server. On Photon OS, kickstart can set the hostname, password, run post-installation scripts, and add public keys for SSH. PXE can be combined with Kickstart to further automate and control the deployment of machines running Photon OS.

For an example of how to deploy Photon OS by using Kickstart, see Photon OS Kickstart Support.

OSTree

OSTree is another option for automating the mass deployment of Photon OS machines. OSTree can deploy and maintain a cluster of Photon OS machines as atomic entities managed through OSTree's imaging capabilities. See the section on OSTree earlier in this paper.

Vagrant

HashiCorp's Vagrant turns a machine's configuration into a distributable template to produce a predictable development environment for applications. Photon OS works seamlessly with Vagrant.

With Vagrant and Photon OS, the vagrant up command can consume a Vagrant box, which is available on the Photon OS GitHub site. You can, for example, create a Photon OS machine running in a VMware Workstation or Fusion or VirtualBox environment, configure it with software by using Chef or Puppet, and use Vagrant to isolate the machine's dependencies and configuration. The Vagrant file containing the machine's template can then be distributed and reused by other developers and testers to reproduce an identical environment.

Photon OS in the Cloud

The mission of Photon OS is to provide an efficient, harmonious Linux platform for emerging cloud technologies. The intention is to spur innovation in cloud-native applications. Photon OS furnishes a ready-made foundation upon which you can build distributed applications that conform to requirements for cloud computing.

The cloud delivers an environment in which you can, through self-service, dynamically adjust compute, networking, and storage resources to match demand. To ease the path to cloud computing, Photon OS comes in several cloud images ready to work with common cloud platforms, including Google Compute Engine (GCE) and Amazon Elastic Cloud Compute (EC2). For a private or hybrid cloud, Photon OS is optimized to run on VMware vSphere and be managed en masse with VMware vCenter®.

By natively working with containers, clustering technologies, and cloud platforms, Photon OS is built with the flexibility to be deployed ubiquitously in the cloud by following standardized deployment methods and by using common components that support cloud-native applications.

The ready-made Photon OS cloud images for GCE, EC2, and vSphere are available for free on the Bintray web site.

VMware vSphere

The minimal version of Photon OS comes in the OVA format to provide a ready-made appliance for running applications in an on-premises cloud based on VMware vSphere with Operations Management™, VMware vCenter, and other VMware software, such as VMware vCloud Suite®.

A vSphere-based private cloud can deliver the following benefits of a public cloud-based service with the security, control, and cost-effectiveness of a fully virtualized platform that pools resources to form an on-premises private cloud:

- Elasticity to dynamically match compute resources with demand
- Secure multi-tenancy to share the infrastructure across lines of business
- Shared virtualization assets to spread cost throughout the organization
- Self-service provisioning to give developers the ability to acquire their own resources through a user portal
- Catalogs of predefined virtual machines, including Photon OS

vSphere maximizes the performance of Photon OS. The operating system is designed to perform optimally when it is running in a VMware cloud that uses ESXi as a host. For more information about how Photon OS is optimized for use with ESXi and vSphere, see the section on performance later in this paper.

Google Compute Engine

Photon OS comes in a preconfigured image ready for Google Cloud Engine (GCE). GCE allows you to run virtual machines on Google's cloud computing infrastructure.

For an operating system to work with GCE, it must match Google's infrastructure requirements. The GCE-ready image of Photon OS contains packages and scripts that prepare it for the Google cloud to save you time as you implement a compute cluster or develop cloud-native applications. For instance, the GCE-ready image of Photon OS includes Google's startup scripts that configure instances, the Google Daemon that creates new accounts and configures SSH, and the Google Cloud SDK that furnishes command-line tools to manage images, instance, and objects on GCE.

By providing a GCE-ready image of Photon OS, VMware streamlines the production process for developing and deploying cloud-native applications on Google's cloud infrastructure. You can, quite quickly, deploy a compute cluster of Photon OS machines running Docker images to deliver, for instance, a mobile application.

Amazon Elastic Cloud Compute

Photon OS comes in an Amazon machine image ready-made for Amazon Elastic Compute Cloud (EC2). The AMI version of Photon OS is available as a free download on Bintray.

You can customize the Photon OS machine by using cloud-init with an EC2 data source. The cloud-init service is commonly used on EC2 to configure the cloud instance of a Linux image. On EC2, for example, cloud-init sets the .ssh/authorized_keys file to let you log in with a private key.

For an example of how to use cloud-init to customize Photon OS in the Amazon cloud, see the section titled Customizing a Photon OS Machine on EC2 in the Photon OS Administration Guide.

Cloud-Init

Although you can run Photon OS on a cloud platform such as GCE or EC2 without using cloud-init, it lets you automate commands that might otherwise require manual intervention.

The minimal and full versions of Photon OS include the cloud-init service as a built-in component. Cloud-init is a set of Python scripts that initialize cloud instances of Linux machines.

The cloud-init scripts configure SSH keys and run commands to customize the machine without user interaction. The commands can set the root password, create a hostname, configure networking, write files to disk, upgrade packages, run custom scripts, and restart the system.

There are several ways in which you can deploy Photon OS with cloud-init, including the following:

- As a stand-alone Photon OS machine
- In Amazon Elastic Compute Cloud (EC2)
- In the Google cloud through the Google Compute Engine, or GCE
- In a VMware vSphere on-premise cloud

When a cloud instance of Photon OS starts, cloud-init requires a data source. The data source can be an EC2 file for Amazon's cloud platform, a seed.iso for a stand-alone instance of Photon OS, or the internal capabilities of a system for managing virtual machines, such as VMware vSphere or vCenter. Cloud-init also includes data sources for OpenStack, Apache CloudStack, and OVF.

On Photon OS, cloud-init is enabled and running by default.

Clustering Technologies on Photon OS

Photon OS works with open source clustering technologies to empower you to rapidly develop distributed cloud-native applications and deploy them at scale in a standard, uniform way on the infrastructure of your choice.

Docker Swarm

Photon OS works with Docker Swarm to turn a collection of Docker engines into a single, distributed compute engine with pooled resources. For an example of how to set up Photon OS to work with Docker Swarm, see Install and Configure a Swarm Cluster with DNS Service on Photon OS.

Kubernetes

Photon OS includes Kubernetes to automatically deploy the containers running a distributed application into a single, manageable entity. As a result, you can horizontally scale the application and orchestrate its containers to meet industrial-strength requirements.

For many teams developing distributed cloud-native applications, Kubernetes is a DevOps cornerstone for the scaling and management of containers when the application needs to reach internet scale. By including Kubernetes in its repositories, Photon OS eases the orchestration of a containerized application when it hits production.

Another advantage of running Kubernetes on Photon OS is that it makes it easier to move a containerized workload to the production environment that is most cost-effective for you, whether it be on-premises, in a hybrid cloud, or in a public cloud.

etcd

etcd is an open source, distributed key-value store that can hold the configuration and state of a cluster. Often coupled with Kubernetes to store data for a distributed, containerized application, etcd lets an application monitor the values in its key-value pairs so that the application can dynamically reconfigure itself when the values change.

In the minimal version of Photon OS, etcd is included in the repository so it can be installed on demand. In the full version of Photon OS, etcd is installed and running by default.

Mesos

Photon OS also includes Apache Mesos to turn your data center's computers, whether virtual or physical, into a pool of resources that runs a distributed application in a fault-tolerant way. By abstracting the CPUs, storage capacity, memory, and other aspects of your data center into a pool of resources that you can tailor to your application, you get elasticity—the ability to dynamically match compute resources to demand.

Performance Optimization for ESXi

A central motivation behind the development of Photon OS is producing a minimalist operating system that delivers performance in a cloud-native stack.

The performance gains from Photon OS are particularly acute when Photon OS runs on VMware ESXi, which can be part of either Photon Platform or a vSphere private cloud.

In the version of Photon OS that comes in the OVA format, the Linux kernel is optimized to run in VMware ESXi. The package name of the kernel is linux-esx. Paravirtualized features, improved boot times, enhanced run-time performance, and general kernel optimizations turn Photon OS into a hyperfast virtual machine on the ESXi hypervisor. Photon OS is also built with open-vm-tools, which makes it optimized for ESXi out of the box.

To achieve these gains in performance, VMware engineers modified Linux kernel configurations, boot parameters, and kernel source code. This section presents an overview of these modifications and briefly discusses their effect on performance when the OVA version of Photon OS runs in ESXi.

Optimizations Using Paravirtualization

Paravirtualization, or PV, optimizes Photon OS for ESXi by making the operating system virtualization-aware. The paravirtualized features in Photon OS include stolen time accounting, the PV scheduler clock, and the PV clocksource.

Stolen time is defined as the time a vCPU would have spent running guest code, but was unable to because CPU resources were unavailable. Stolen time accounting keeps track of this time to improve process accounting and scheduler fairness in a guest.

Photon OS implements a PV sched_clock to deliver accurate kernel boot time reporting by eliminating time drifting during boot process. The PV scheduler clock also reduces context switching overhead to improve performance.

The paravirtualized version of the clock source in Photon OS takes priority during the boot process to produce faster boot times.

Boot Time Improvements

Boot time improvements stem from removing all device drivers and kernel subsystems that go unused when Photon OS runs in ESXi—for example, power management, suspend to RAM, hibernate, CPU frequency scaling, and most of the ACPI features.

Photon OS also refrains from probing known PCI devices, instead scanning the list of existing devices. In addition, the asynchronous root file system mount does not wait for known devices to complete their probing.

Photon OS removes unnecessary timeouts dictated by hardware and neither reseeds nor verifies RDRAND instructions when ESXi supports it.

Finally, some device drivers, such as the vmxnet3 serial port, are built as modules instead of as part of the kernel so that they load in parallel during system boot.

General Kernel Tuning

General kernel optimizations include changes to the default settings for kernel preemption, TTY-based group scheduling, timer tick handling, transparent huge page, and the I/O scheduler.

Kernel preemption is the ability to schedule out a task while it is running in kernel mode. Although kernel preemption improves response time, it affects throughput and slows down critical paths in the Linux kernel because of the increased overhead. Since preemption is typically discouraged in server workloads, Photon OS disables preemption using the kernel configuration option set as PREEMPT=n.

Another feature that can dramatically impact forking servers is task auto-grouping. It groups tasks by TTY, to improve perceived responsiveness on an interactive system. On a server with a long running forking daemon, this will tend to keep child processes from migrating away as soon as they should. Photon OS disables the SCHED_AUTOGROUP option in the kernel configuration to improve performance.

Photon OS uses the idle tickless mode with the timer frequency scaled down to 250Hz to improve performance in virtualized environments.

Photon OS updates the TRANSPARENT_HUGEPAGE kernel option so it is always enabled. The transparent huge page kernel option automatically starts using large pages for dynamically allocated memory to enhance performance by reducing the pressure on the system's translation look-aside buffer (TLB), which speeds up memory access when Photon runs in a hypervisor.

VMware ESXi uses an asynchronous intelligent IO scheduler. Since ESXi does its own sorting, into its own disk queues, there's little point in a guest OS attempting the same work—it can introduce latency, waste some CPU cycles, or slow down IO operations between the hypervisor and its back-end storage. Photon OS uses the NOOP IO scheduler to add IO requests to a FIFO queue and assumes IO performance optimization will be handled by the hypervisor, boosting run-time performance.

Photon OS in Photon Platform

Photon OS plays a role in VMware Photon Platform. Purpose-built to develop, run, and automate cloud-native applications on virtualized infrastructure, Photon Platform combines several VMware components into an IaaS solution optimized for API-driven, multitenant, high-scale environments. It supports such next-generation frameworks as Docker, Pivotal Cloud Foundry, and Kubernetes.

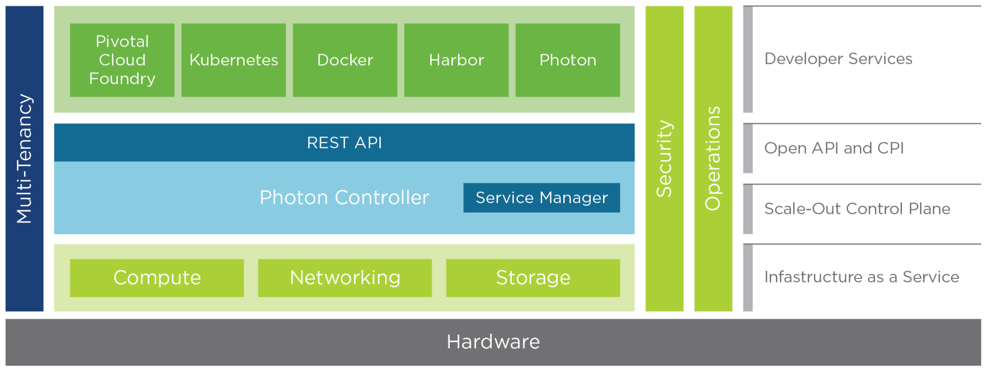
The platform combines VMware Photon™ Controller with Project Lightwave, VMware NSX®, and VMware vSAN™. Photon Controller manages virtualized infrastructure and delivers Kubernetes as a service to securely run containerized workloads at scale. Lightwave secures Photon Platform, NSX provides software-defined virtualized networking, and vSAN delivers a virtual storage area network.

The architecture of Photon Platform revolves around Photon Controller, which provides the core services that enable system administrators and DevOps to set up tenants, projects, virtual machines, Docker containers, Kubernetes clusters, and other infrastructure in a multitenant environment secured by Lightwave. Developers and DevOps managers can interact with Photon Controller as tenants to create projects for developing and deploying cloud-native applications.

In Photon Platform, Photon OS can serve as the operating system for virtual machines running in ESXi, furnishing simple, stackable, replaceable hosts for containerized applications. Kubernetes clusters use Photon OS for their nodes.

Here's a high-level view of the architecture of Photon Platform:

VMware Photon Platform



For more information, see VMware Cloud Native Applications.

Conclusion

Photon OS is a minimalist Linux operating system optimized for developing cloud-native applications, which tend to be composed of microservices, packaged in containers, and scheduled to run dynamically in nodes on distributed infrastructure at scale.

By integrating seamlessly with the VMware ecosystem—including VMware vSphere and Photon Platform—Photon OS delivers a ready-made foundation for rapidly building and deploying cloud-native applications while continuing to fulfill such IT requirements as cost-effectiveness, performance, and security.