WHITE PAPER – **AUGUST 2017**

DRIVING DIGITAL TRANSFORMATION WITH CONTAINERS AND KUBERNETES

How Kubernetes Manages Containerized Applications to Deliver Business Value

vmware®

Table of Contents

DOCKER CONTAINER DEFINED

With containers, Docker has defined a standard format for packaging and porting software, much like ISO containers define a standard for shipping freight. As a runtime instance of a Docker image, a container consists of three parts:

- A Docker image
- An environment in which the image is executed
- A set of instructions for running the image

—Adapted from the Docker Glossary

Introduction

Kubernetes manages containers. Containers package applications and their dependencies into a distributable image that can run almost anywhere, streamlining the development and deployment of software. By adopting containers, organizations can take a vital step toward transforming themselves into agile digital enterprises focused on accelerating the delivery of innovative products, services, and customer experiences. Enterprises can become the disrupters instead of the disrupted.

But containers create technology management problems of their own, especially when containerized applications need to be deployed and managed at scale, and that's when Kubernetes comes into play. Kubernetes orchestrates containerized applications to manage and automate resource utilization, failure handling, availability, configuration, scalability, and desired state.

This paper describes Kubernetes, explains its business value, explores its use cases, and illuminates how it can accelerate your organization's digital transformation.

The Digital Transformation and the Shift to Containerized Applications

The rate of technological innovation is, according to the The New York Times, increasing and expanding.[1] In response, digital transformation has become an established objective for many enterprises, and the adoption of digital initiatives is now widespread.[2]

The reasons enterprises are undergoing digital transformation are clear:

- Create new applications that engage customers in innovative and captivating ways.
- Improve operations to more efficiently deliver better products and services at a lower cost to the business.
- Generate new revenue streams by rapidly adapting to changes in market conditions and consumer preferences.

"The future," Gartner Research says, "will belong to companies that can create the most effective, smart and autonomous software solutions."[3] But the ingredients for building effective, autonomous applications are less clear than the desired outcomes.

To be effective in this era, applications require an architecture that fosters fluid, rapid, responsive development and deployment while still maintaining the security, performance, and cost-effectiveness of established patterns. Containers provide the basis for a new application architecture that supports digital transformation and lays the foundation for innovation.

1 "Digital Transformation Going Mainstream in 2016, IDC Predicts", Steve Lohr, The New York Times, November 4, 2015.

2 "New Research Finds Investment from Outside IT Is Key to Digital Transformation Success", from the Business Wire, The New York Times, May 11, 2017.

3 "Digital Transformation Going Mainstream in 2016, IDC Predicts", Steve Lohr, The New York Times, November 4, 2015.

Enterprises are increasingly adopting container technology. A recent survey by 451 Research revealed a profile of impressive implementation for an emerging ecosystem.[4] Organizations that are adopting containers see them as a fast track to building and deploying cloud-native applications and twelve-factor apps.

Cloud-Native Applications and 12-Factor Apps

The Cloud Native Computing Foundation, a project of The Linux Foundation, defines cloud-native applications as follows:[5]

1. **Containerized**—Each part (applications, processes, etc.) is packaged in its own container. This facilitates reproducibility, transparency, and resource isolation.
2. **Dynamically orchestrated**—Containers are actively scheduled and managed to optimize resource utilization.
3. **Microservices oriented**—Applications are segmented into microservices. This segmentation significantly increases the overall agility and maintainability of applications.

Kubernetes covers the second part of the definition by scheduling and managing containers. For the third part, both Kubernetes and Docker help implement microservices.

The key element, however, is the container—a process that runs on a computer or virtual machine with its own isolated, self-described application, file system, and networking. A container packages an application in a reproducible way: It can be distributed and reused with minimal effort.

Docker containers are the most widely deployed container. A manifest, called a Dockerfile, describes how the image and its parts are to run in a container on a host. To make the relationship between the Dockerfile and the image concrete, here's an example of a Dockerfile that installs MongoDB on an Ubuntu machine running in a container. The lines starting with a number sign are comments describing the subsequent commands.

```
# MongoDB Dockerfile from https://github.com/dockerfile/mongodb
# Pull base image.
FROM dockerfile/ubuntu
# Install MongoDB.
RUN \
  apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
7F0CEB10 && \
  echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-
upstart dist 10gen' > /etc/apt/sources.list.d/mongodb.list && \
  apt-get update && \
  apt-get install -y mongodb-org && \
  rm -rf /var/lib/apt/lists/*
# Define mountable directories.
```

4 "Application containers will be a $2.7bn market by 2020, representing a small but high-growth segment of the CloudEnabling Technologies market", 451 Research, Jan. 10, 2017.

5 This definition is from the FAQ of the Cloud Native Computing Foundation, https://www.cncf.io/about/faq/.

```
VOLUME ["/data/db"]
# Define working directory.

WORKDIR /data
# Define default command.
CMD ["mongod"]
# Expose port 27017 for the process and port 28017 for http
EXPOSE 27017
EXPOSE 28017
```

Methodology for Delivering Software as a Service

In contrast, the 12-factor app is defined as much by its processes as by its systemic properties. It is a methodology for developing a software-as-a-service (SaaS) application—that is, a web app—and typically deploying it on a platform-as-a-service (PaaS), such as Pivotal Cloud Foundry. Here are the 12 factors with a brief explanation of each one:[6]

1. **Deploy the application many times from one codebase.** The codebase is stored in a repository, managed with a version control system such as Git as it is modified, and then deployed many times as a running instance of the app from that the same codebase. As a result, a deployment is often running in three environments: on each developer's local environment, in a staging environment, and in the production environment.
2. **Declare and isolate dependencies.** The app does not implicitly rely on system-wide packages; instead, it declares the dependencies in a declaration manifest. Explicitly declaring dependencies makes it easier for new developers to set up their development environment.
3. **Store the configuration in the environment, not the code.** For configuration information that varies by deployment, the app stores the information in environmental variables. The environmental variables are granular controls that are managed independently for each deployment so that the app can easily scale into more deployments over time.
4. **Connect to supporting services**, such as a database or a storage system, instead of including it in the code. The app treats such services as resources that can be attached to or detached from a deployment by modifying the configuration.
5. **Treat build and run as separate stages.** A deployment of the codebase takes place in three separate stages: build, release, and runtime. The build stage converts the codebase into an executable—a build— and then the release stage combines the build with the configuration to produce a release that's ready for execution in the runtime environment.
6. **Run the app as stateless processes.** The processes share nothing with other processes, and data that must persist is stored in a database running as a stateful supporting service.

6 The twelve factors are paraphrased from the descriptions at the Twelve-Factor App web site.

MANAGING CONTAINERIZED APPLICATIONS WITH KUBERNETES

Kubernetes orchestrates distributed, containerized applications to:

- Optimize utilization of computing resources.
- Provide policies for scheduling.
- Maintain desired state.
- Handle faults and failures with automation.
- Provide high availability.
- Monitor jobs in real-time.
- Manage an application's configuration.
- Dynamically scale to meet changes in demand.

7. **Expose services by using port binding.** Taking HTTP as an example, the app exports HTTP as a service by binding to a port and listening on the port for incoming requests.
8. **Scale out by adding concurrent processes.** The app handles workloads by assigning each type of work to a process type. A web process, for example, handles HTTP requests, while a worker process manages background tasks.
9. **Ensure durability with disposability.** Processes are disposable—they can be started or stopped quickly to make sure that the application can be changed or scaled easily.
10. **Make development and production peers.** The app is geared toward continuous deployment by allowing developers to integrate new code quickly and to deploy the app themselves in a production environment. The production and development environments should be as similar as possible.
11. **Process logs as event streams.** The app neither routes nor stores the output stream from its logs but instead writes it as a stream of data to standard output, where it is to be collected by the execution environment and routed to a tool or system, such as Hadoop, for storage or analysis.
12. **Run one-off management scripts and tasks**, such as a database migration, in an environment identical to that of the app's long-running processes.

Containers and Kubernetes help satisfy aspects of these imperatives. Containers, for example, play a key role in 12-factor apps by letting you declare and isolate dependencies. Containers also help ensure durability with disposability by, among other things, starting quickly and stopping gracefully. Many of the other factors are supported by Kubernetes.

A Concise Overview of Kubernetes

Google originally developed Kubernetes. The company uses its predecessor, called Borg, to initiate, schedule, restart, and monitor public-facing applications, such as Gmail and Google Docs, as well as internal frameworks, such as MapReduce.[7] Based on Google's original system plus enhancements from the lessons learned with Borg, Kubernetes is an open source orchestration system for containers that can work in your data center, across clouds, and in a hybrid data center. Kubernetes automatically places workloads, restarts applications, and adds resources to meet demand.

Here, briefly, is how it works. A Kubernetes cluster contains a master node and several worker nodes. Then, when you deploy an application on the cluster, the components of the application run on the worker nodes. The master node manages the deployment.

Kubernetes includes these components:

- The Kubernetes API
- The Kubernetes command-line interface, kubectl
- The Kubernetes control plane

7 Large-Scale Cluster Management at Google with Borg, Research at Google, 2015.

ADVANTAGES OF USING KUBERNETES

- Consolidate servers and reduce costs through efficient resource utilization.
- Elegantly handle machine failure through self-healing and high availability.
- Ease and expedite application deployment, logging, and monitoring.
- Automate scalability for containers and containerized applications.
- Decouple applications from machines for portability and flexibility.
- Easily modify, update, extend, or redeploy applications without affecting other workloads.

The control plane comprises the processes running on the Kubernetes master and on each worker node. On the master, for example, Kubernetes runs several processes: the API server, the controller, the scheduler, and etcd. The worker nodes run the "kubelet" process to communicate with the master and the proxy process to manage networking.

Kubernetes Object Model

One of the keys to the Kubernetes system is how it represents the state of the containerized applications and workloads that have been deployed. Kubernetes represents state by using "objects," such as service, namespace, and volume. These objects are typically set by an object specification, or spec, that you create for your cluster.

In the Kubernetes object mode, the concept of a Pod is the most basic deployable building block. A Pod represents an instance of an app running as a process on a Kubernetes cluster. Here's where the Docker runtime comes back into the equation—Docker is commonly used as the runtime in a Kubernetes Pod.

Kubernetes also includes Controllers that implement most of the logic in Kubernetes. The Controllers provide features such as the replica set and the stateful set.

Maintaining the Desired State

The Kubernetes control plane manages the state of all these objects to ensure that they match your desired state. You can specify a desired state by creating an object specification for a service with a YAML file. Here's an example:

```
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: myrepo/nginx
  ports:
   - containerPort: 80
```

When you submit this file to the Kubernetes master with the kubectl command-line interface, the Kubernetes control plane implements the instructions in the file by starting and scheduling applications so that the cluster's state matches your desired state. The Kubernetes master and the control plane then maintain the desired state by orchestrating the cluster's nodes, which can be actual servers or virtual machines.

The core of the architecture is an API server that manages the state of the system's objects. The API server works with Kubernetes subcomponents, or clients, that are built as composable microservices, such as the replication controller specified in the YAML file. The replication controller regulates the desired state of pod replicas when failures occur.

Business Value of Kubernetes

Returning to the digital transformation, Kubernetes uses this architecture to manage containerized applications in a distributed cluster. The results help fulfill the business promise of digital transformation:

- Kubernetes makes it easier and cheaper to run applications in public, private, or hybrid clouds.
- Kubernetes accelerates application development and deployment.
- Kubernetes increases agility, flexibility, and the ability to adapt to change.

Kubernetes for Cloud-Native and 12-Factor Applications

Kubernetes makes containerized applications work in a manageable way at scale. Recall the second part of the definition of cloud-native applications: They are dynamically orchestrated in such a way that containers are actively scheduled and managed to optimize resource utilization. Kubernetes does exactly that. It orchestrates containers and their workloads to optimize the utilization of the virtual machines and physical servers that make up the nodes in a cluster.

Revisiting the 12 factors further illustrates how Kubernetes streamlines application management. In general, Kubernetes can deploy and run 12-factor apps.

HOW KUBERNETES AND CONTAINERS STREAMLINE APPLICATION MANAGEMENT	
Factor	**Benefit**
1. Deploy the application many times from one codebase.	Kubernetes can deploy applications with one code base many times by giving a pod a specification that includes a container image reference.
2. Declare and isolate dependencies.	Containers can express dependencies.
3. Store the configuration in the environment, not the code.	You can store aspects of an application's configuration in the Kubernetes environment. For example, the ConfigMaps construct separates configuration artifacts from an image's instructions.
4. Connect to supporting services, such as a database, instead of including it in the code.	Kubernetes lets you deploy supporting services, such as a database, in separate containers and then manages all the containerized components together to ensure availability and performance.
5. Treat build and run as separate stages.	You can, for example, build the application by using Jenkins (a pipeline automation server separate from Kubernetes) and then run the Docker images by using Kubernetes.
6. Run the app as stateless processes.	Kubernetes makes it easy to run stateless applications. Kubernetes allows states to be maintained independently in an etcd data store, for instance, while the application runs. Kubernetes also allows you to attach persistent storage. The spec file defining a Pod, for example, can require a persistent volume; if the Pod goes down, the replacement Pod connects to the same persistent volume.
7. Expose services by using port binding.	Kubernetes includes configuration options for exposing services on ports. In the nginx example YAML file that appeared earlier, the nginx web server was bound to Port 80 and exposed as a service.
8. Scale out by adding concurrent processes.	Kubernetes scales an application by adding more Pods. Kubernetes can use the replication controller, for example, to add multiple Pods at the same time.
9. Ensure durability with disposability.	Containers running in Kubernetes are seen as mutable—they are to be stopped and replaced on demand or on a schedule.
10. Make development and production peers.	The Kubernetes environment lets development and production code be rigorously tested in the same way. For instance, you can use a Kubernetes deployment with two pods, one pod that contains the production environment and another pod that contains the staging environment, which in effect makes staging and production peers. In addition, the environment specified in a container is uniform across development and production environments.
11. Process logs as event streams.	Kubernetes lets you access the standard output of a container so that you can process its output as a data stream with the tool of your choice, such as VMware vRealize® Log Insight™.
12. Run management tasks as one-off processes.	You can schedule a Pod consisting of the application container using a different entry point to run a different process, such as a script to migrate a database.

BENEFITS FOR DEVELOPERS

The business value of containers and Kubernetes isn't limited to the business as a whole or the office of the CIO. Developers like containers because they make life easier, development more engaging, and work more productive.

- **Portability:** Containers let developers choose how and where to deploy an app.
- **Speed:** Containers expedite workflows like testing and speed up iterations.
- **CI/CD Pipeline:** Kubernetes and containers support continuous integration and continuous deployment.
- **Flexibility:** Developers can code on their laptops when and where they want with the tools they like.
- **The 13th Factor:** Containers and Kubernetes are seen as fashionable technologies. Developers are highly motivated to use them.

An Example Use Case

A short case study provides a high-level use case for managing containers with Kubernetes.

A taxicab company in a major metropolitan area is losing riders to car-sharing services, imperiling its once-strong local market share. It needs to transform itself into a digital enterprise capable of competing with car-sharing companies. To do so, the company wants to develop its own mobile app, cost-effectively run the app in its modest data center, and attempt to provide innovative services.

To its credit, the taxi company retains a number of advantages: a well-known, long-established local brand with a reputation for timely, courteous, safe drivers.

As recently hired developers work on the mobile app, the taxi company modernizes its data center with commodity hardware and virtualization. To maximize resource utilization of its small data center and to minimize costs, the company plans to run its new app in Docker containers on virtual machines. Kubernetes will orchestrate the containerized application.

After being rolled out and advertised in and on its cars, the app is an instant success. To meet fluctuations in use of the app, the company uses Kubernetes to dynamically scale the number of containers running the app. For example, when metrics for the app hit a predefined threshold indicating high usage, which typically happens during rush hour, the company's DevOps team uses the horizontal pod autoscaling feature of Kubernetes to automatically maximize the number of containers so that the system can match demand. At 4 am, in contrast, the number of containers is reduced to elastically match the low demand at that time, conserving resources.

The mobile app correlates ride requests with location. By mining the data and combining it with its intimate historic knowledge of the city's patterns, the cab company can station cabs in the perfect locations for hailing customers—preempting some car requests to the competition. What's more, because the company processes the app's logs as event streams, the company can do this dynamically during day and night, shifting cars to hot spots.

Because the company implemented the app by using containers, developers can roll out new changes daily. The data that the app collects helps the company pinpoint new features and quickly innovate to focus on its strengths, such as identifying recurring customers and rolling out a rewards program to retain them.

The business benefits of the company's technical agility, containerized application, and Kubernetes orchestration add up to a competitive advantage:

- The scheduling policies in Kubernetes give the company the elasticity it needs to dynamically match demand in a cost-effective way with its modest but now-modernized data center.
- Faults and failures are handled automatically by Kubernetes, reducing troubleshooting demands on its small DevOps staff.
- The seamless modification of the app and its features helps the company beat its bigger, less local rivals by being more agile and better able to apply its knowledge of local patterns.

- Containers and Kubernetes make it easier and cheaper to run the app.
- The ease with which the DevOps team can port containers from the test environment to production accelerates the development and deployment of new features.

Container Technology Solutions from VMware

In a recent report titled "Closing the Digital Transformation Confidence Gap in 2017," The Hackett Group surveyed executives from more than 180 large companies. The report found a wide confidence gap "between the high expectations for digital transformation's business impact and the low perception of the business's capability to execute digital transformation." The Hackett group says that the findings demonstrate the need for IT to invest in the necessary tools and to adopt rapid application development techniques, such as agile processes.[8]

Cloud-native solutions from VMware help you quickly and cost-effectively put containers into production, improving your ability to carry out digital transformation.

Running containers on VMs also adds a beneficial level of security to containerized applications, especially in the context of the third tenet of cloud-native applications—microservices. According to a Docker white paper on security, "Deploying Docker containers in conjunction with VMs allows an entire group of services to be isolated from each other and then grouped inside of a virtual machine host."[9]

Deploying containers with VMs encases an application with two layers of isolation, an approach that is well-suited to cloud-style environments with multitenancy and multiple workloads. "Docker containers pair well with virtualization technologies by protecting the virtual machine itself and providing defense in-depth for the host," the Docker security white paper says.

vSphere Integrated Containers

A comprehensive container solution built on VMware vSphere, VMware vSphere Integrated Containers runs modern and traditional workloads side by side in your VMware software-defined data center with enterprise-grade networking, storage, security, performance, and visibility.

With support for Docker containers, vSphere Integrated Containers empowers you to immediately use container technology to enhance developer productivity and business agility. The solution helps transform your organization into a digital enterprise and to modernize your data center by deploying containerized applications.

8 Despite High Expectations for Digital Transformation Led by Cloud, Analytics, Robotic Process Automation, Cognitive & Mobile, IT & Other Business Services Areas See Low Capability to Execute, The Hackett Group, March 16, 2017. A version of the research is available for download, following registration, at http://www.thehackettgroup.com/research/2017/social-media/key17it/.

9 Introduction to Container Security, Docker white paper, Docker.com.

Architecture Overview

The following diagram illustrates the high-level architecture of vSphere Integrated Containers.

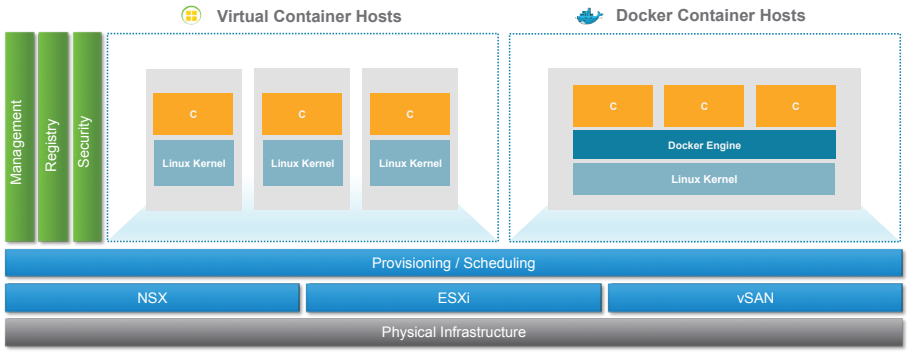

Figure 1: High-levelarchitecture of vSphere Integrated Containers

This architecture gives you two container deployment models:

- **Virtual container hosts:** vSphere Integrated Containers takes advantage of the native constructs of vSphere to provision containers. By deploying each container image as a virtual machine, vSphere Integrated Containers extends the availability and performance features of vSphere to containerized workloads, including VMware HA, vMotion and Distributed Resource Scheduler. In addition, developers can consume a Docker API.
- **Docker container hosts:** Developers can self-provision native Docker container hosts on demand and run them on vSphere. The ticketless environment that vSphere Integrated Containers provides lets developers use Docker tools while giving IT teams governance and control over the infrastructure.

Components

Each of the components of vSphere Integrated Containers is an open source project:

- **vSphere Integrated Containers Engine:** A container runtime for vSphere, the engine enables software engineers to develop in containers and deploy containerized apps alongside traditional VM-based workloads on vSphere clusters.
- **Harbor:** An enterprise-class private container registry that stores and distributes container images, Harbor extends the open source distribution of Docker with such enterprise functionality as identity management, role-based access control, and auditing.
- **Admiral:** A container management portal, Admiral supplies a user interface for DevOps teams and others to provision and manage containers. Admiral can, for instance, show metrics about container instances. Cloud administrators can manage container hosts and apply governance to their usage, including capacity quotas.

THE BENEFITS OF MICROSERVICES

Coupled with containers, microservices are increasingly becoming the architectural pattern of choice for developing a new application. The architecture breaks up the functions of an application into a set of small, discrete, decentralized, goal-oriented processes, each of which can be independently developed, tested, deployed, replaced, and scaled.

- Increase modularity
- Make app easier to develop and test
- Parallelize development: A team can develop and deploy a service independently of other teams working on other services
- Support continuous code refactoring to heighten the benefits of microservices over time
- Drive a model of continuous integration and continuous deployment
- Improve scalability
- Simplify component upgrades

Capabilities

vSphere Integrated Containers includes a unified management portal that is integrated with identity management to securely provision containers. Developers and DevOps can serve their own requirements by creating Docker container hosts on demand.

The result enables application development teams to build, test, and deploy containerized applications. The solution supports agile development practices and DevOps methodologies like continuous integration and continuous deployment (CI/DI).

Wavefront by VMware

Wavefront® by VMware efficiently monitors containers at scale. The Wavefront platform includes dashboards that give DevOps real-time visibility into the operations and performance of containerized applications and Kubernetes clusters.

The Wavefront service can measure, correlate, and analyze data across containers and Kubernetes clusters. The dashboard displays data on the performance of microservices and resource utilization to help you identify issues and optimize applications. The data can, for example, help make decisions about how and when to scale a container environment. For more information, see VMware and Wavefront.

Pivotal Container Service

VMware® Pivotal Container Service offers production-grade Kubernetes for enterprises to reliably deploy, run, and operationalize modern and traditional applications across private and public clouds. Based on the open source project Kubo, Pivotal Container Service delivers high availability, advanced security, and operational efficiency for enterprises to shorten time to market, increase developer productivity, and lower operating expenses.

To provide a fast path to production for microservices and containerized workloads, Pivotal Container Service establishes a unified virtualization and container infrastructure on VMware vSphere or in a VMware software-defined data center.

Components

Pivotal Container Service includes the following components:

- **Production-grade Kubernetes.**
- **BOSH.** It is an open source system that unifies release engineering, deployment, and lifecycle management for small- and large-scale cloud software. Well-suited to large distributed systems, the system performs monitoring, failure recovery, and software updates with zero-to-minimal downtime. BOSH supports multiple infrastructure-as-a-service providers, including VMware vSphere, Google Cloud Platform, Amazon Amazon Elastic Compute Cloud (EC2), and OpenStack.

LEARN MORE ABOUT ...

To learn how solutions from VMware can help you build, run, and manage cloud-native applications, visit: cloud.vmware.com/cloud-native-apps

- **VMware ESXi™.** The industry-leading, purpose-built bare-metal hypervisor, ESXi installs directly onto your physical server, enabling it to be partitioned into virtual machines.
- **VMware NSX®.** This network virtualization technology for modern application architectures provides key networking capabilities to Kubernetes clusters.

Conclusion

Building and deploying containerized applications on VMware infrastructure drives business value through digital transformation. VMware solutions enhance developer productivity, business agility, IT flexibility, and application scalability. The result helps you adapt to changes in the marketplace and shorten the time it takes to bring an application to market.

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com