WHITE PAPER - **OCTOBER 2017**

CONTAINER PLATFORMS FOR APP MODERNIZATION

Matching Use Cases with Container Technology

Table of Contents

Introduction

This paper describes the basics of container technology, explains the properties of three elemental container platforms, and maps different use cases to the three platforms. The objective is to help you understand container platforms, their underlying technology, and their business value so that you can make an informed decision about the best platform for your organization, its use cases, and its goals.

Container Technology and Terminology

Container technology comes with its own lexicon. The discussion about container platforms later in this paper assumes that you understand the following terminology. If you're familiar with containers, Kubernetes, and cloud-native applications you can skip this section.

Containers

Container: A portable format, known as an *image*, for packaging an application with instructions on how to run it as well as an environment in which the image is executed. When the container image is executed, it runs as a process on a computer or virtual machine with its own isolated, self-described application, file system, and networking. The use of containers is increasing because they provide a portable, flexible, and predictable way of packaging, distributing, modifying, testing, and running applications. Containers speed up software development and deployment.

Docker is a widely used container format. Docker defines 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

Containerized application: An application that has been packaged to run in a container.

Kubernetes and Orchestration

Kubernetes: A system that automates the deployment and management of containerized applications. As an application and its services run in containers on a distributed cluster of virtual or physical machines, Kubernetes manages all the moving pieces to optimize the use of computing resources, to maintain the desired state, and to scale on demand. On Kubernetes, a container (or a set of related containers) is deployed in a logical unit called a *pod*. In addition to scheduling the deployment and automating the management of containerized applications, a key benefit of

ADVANTAGES OF KUBERNETES

- Consolidate servers and reduce costs through efficient resource utilization..
- Ease and expedite application deployment.
- Decouple applications from machines for portability and flexibility.
- Easily modify, update, extend, or redeploy applications without affecting other workloads.
- Handle faults and failures with automation.
- Dynamically scale to meet changes in demand.

Kubernetes is that it maintains the *desired state*—the state that an administrator specifies the application should be in.

Cluster: Three or more interconnected virtual machines or physical computers that, in effect, form a single system. A computer in a cluster is referred to as a node. An application running on a cluster is typically a *distributed application* because it runs on multiple nodes. By inherently providing high availability, fault tolerance, and scalability, clusters are a key part of cloud computing.

Orchestration: Because it can automatically deploy, manage, and scale a containerized application, Kubernetes is often referred to as an orchestration framework or an orchestration engine. It orchestrates resource utilization, failure handling, availability, configuration, desired state, and scalability.

Application Types and Architectural Patterns

Microservices: A "modern" architectural pattern for building an application. A micro-services 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.

Cloud-native applications: Generally speaking, they are developed and optimized to run in a cloud as distributed applications. According to the Cloud Native Computing Foundation, cloud-native applications, which are also generally referred to as "modern" applications, are marked by the following characteristics:

- Containerized for reproducibility, transparency, and resource isolation.
- Orchestrated to optimize resource utilization.
- Segmented into microservices to ease modification, maintenance, and scalability.

Cloud-native applications are typically developed and deployed on a containers as a service platform (CaaS) or a platform as a service.

12-factor app: 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) or a containers as a service (CaaS).

Platforms

The overarching business objective of using a container platform is to accelerate the development and deployment of scalable, enterprise-grade software that is easy to modify, extend, operate, and maintain. Three types of platforms provide varying degrees of support for container technology:

vmware®

- A platform for running individual container instances.
- Containers as a service.
- Platform as a service.

A platform as a service is often referred to simply as an *application platform*. In this context, an application platform helps developers not only write code but also integrate tools and services, such as a database, with their application as microservices. An example of a private platform as a service that is also referred to as an application platform is Pivotal Cloud Foundry.

A container-as-a service platform helps developers build, deploy, and manage containerized applications, typically by using Kubernetes or another orchestration framework. An example of a container as a service platform is VMware Pivotal Container Service.

A platform for running container instances helps developers build and test a containerized application. It does not, however, orchestrate the containerized application with Kubernetes, nor does it provide a service broker so that developers can integrate tools, databases, and services with an app. An example of a container instance platform is VMware vSphere Integrated Containers.

A later section matches different types of applications, use cases, and business objectives with these three types of container platforms.

Platforms and Developer Operations

Delivering software in an expedient, reliable, sustainable way requires collaboration between IT teams and developers. *DevOps* takes place when developers and IT come together to focus on operations in the name of streamlining and automating development and deployment. DevOps is key practice driving cloud-native applications.

To help DevOps, a container platform provides some or all of the following services:

- Lets developers add tools and services to their app through a service broker or catalog.
- Adds security, logging, monitoring, analytics, dashboards, maintenance, and other operational features.
- Provides container networking.
- Exposes an API.
- Automates delivery and deployment processes.
- Eases continuous integration, continuous delivery, and continuous deployment.

THE BENEFITS OF MODERNIZING

- Faster development and deployment
- Agility
- Flexibility
- Scalability
- Portability
- Process automation
- Resource optimization
- Easier maintenance
- Automated operations

Continuous Integration, Delivery, and Deployment

Continuous integration constantly combines source code from different developers or teams into an app and then tests it. *Continuous delivery* readies an application or part of an application for production by packaging and validating it. *Continuous deployment* automatically deploys an application or part of an application into production. The entire process forms the CI/CD pipeline when the *D* in the abbreviation is assumed to represent deployment.

Digital Transformation

All the modernizing elements covered in this section—containers, Kubernetes, microservices, container platforms, DevOps, and the CI/CD pipeline—converge into a powerful recipe for digital transformation: You can optimize the use of your computing resources and your software development practices to extend your enterprise's adaptability, productivity, innovation, competitive advantage, and global reach.

As you turn to container technology, an effective strategy for digital transformation includes matching the architecture and workloads of your applications to the right platform for the job. Different types of platforms are primed for different levels of container adoption, organizational requirements, and use cases.

High-Level Use Cases for Container Platforms

The following use cases coincide with the extent to which an enterprise has embraced container technology, defining a sort of container-adoption continuum that takes place after an initial stage of experimentation and evaluation:

- Establishing a developer sandbox or self-service agile infrastructure
- Repackaging a legacy application
- Replatforming or migrating a traditional app
- Replatforming and refactoring an app
- Building cloud-native apps or developing on and for the cloud

Maturity of Container Adoption

In the early stages of container adoption, organizations seek ready-to-go development tools and a service portal so that developers can self-service their needs with agile infrastructure.

In the middle stage of the journey, organizations strive to accelerate software development by repackaging traditional applications in containers to simplify developer workflows and application maintenance. This kind of a lift-and-shift use case eventually leads to

FACTORS IN SELECTING A CONTAINER PLATFORM

The platform that you select will depend on your unique situation and goals. Here are some factors to consider:

- Use cases
- Application types and their workloads
- Software development methods and processes
- Operations
- Security and compliance
- Networking
- People and their skill sets
- Maturity of your organization's container adoption
- Maximizing flexibility vs. minimizing complexity
- Business objectives

replatforming the repackaged application so that its deployment can be automated, orchestrated, and scaled on demand.

Cloud Natives

Then there are the cloud natives who live in and build for the cloud. They seek to build new applications by using architectural techniques like microservices and methodologies like the 12-factor app. When it makes sense to do so, they are also working to refactor legacy monolithic applications as cloud-native apps. Organizations at this stage are focused on automation, optimization, and rapid innovation.

Matching the Platform to the Project

Different platforms address different situations, and several factors come into play in analyzing the platform that's right for your stage of container adoption and your use cases:

- Identifying your target use cases or application types and matching them with the best-suited platform—that is, using the right tool for the job.
- Determining how much operational work is to be handled by DevOps or spread among developers and traditional IT teams—that is, having the right workers for the job.
- Deciding how much flexibility you want, including how you handle continuous integration, continuous delivery, and continuous deployment—that is, finding the right balance between prescription and complexity.

Prescription and Complexity

The more prescriptive the platform is, the more it hides the complexity of the platfrom from developers. A prescriptive platform prescribes a scheduler, a runtime engine, integration with the underlying infrastructure, continuous delivery, and other aspects of the platform. (A prescriptive platfrom is also referred to as an opinionated platform.)

For example, a prescriptive platfrom includes its own scheduler for containers and specifies how to use it to run containerized applications. The main advantage of a perscriptive platform is that it places the platform's complexity in a layer of abstraction—all developers have to do is write their code and generate an application artifact, and the platform handles the rest. The disadvantage is that you have fewer options and less flexibility in how you delivery and deploy your app.

Revisiting the Three Platform Types

Recall the three container platforms mentioned earlier:

VSPHERE INTEGRATED CONTAINERS AT-A-GLANCE

vSphere Integrated Containers creates container infrastructure within VMware vSphere. The solution includes an engine to run containers, a portal to manage containers, and a registry to store container images.

By supporting containers in VMware vSphere, IT teams get the security, isolation, and management of virtual machines while developers can exploit the benefits of containers.

KEY BENEFITS OF VSPHERE INTEGRATED CONTAINERS

- Run containerized apps alongside existing workloads
- Combine portability with security, visibility and management
- Provide developers with a Docker-compatible interface

- VMware vSphere Integrated Containers is a platform for running individual instances of containers without orchestration.
- VMware Pivotal Container Service provides containers as a service with Kubernetes orchestration.
- Pivotal Cloud Foundry provides a private platform as a service, also known as a application platform.

vSphere Integrated Containers Use Cases

vSphere Integrated Containers is a flexible platform for running secure container instances. It provides minimal tooling—a container runtime as well as primitives for networking and storage. You must create your own continuous integration, delivery, and deployment systems. If you require a scheduling and orchestration engine, you must set it up.

For the container instance, the developer provides the container image. A traditional IT team might work with DevOps to provide and manage the infrastructure.

This platform's ideal use cases are as follows:

- Hosting long-running containerized services that need strong isolation and high throughput, such as web servers, key value stores, and databases.
- Repackaging a traditional app in a container when it does not warrant container orchestration.
- Providing agile self-service infrastructure to developers, including a developer sandbox, without having to build a dedicated container infrastructure stack.

VMware PKS Use Cases

VMware PKS delivers a highly flexible approach to working with containers. Developers provide container images and templates for pods.

At the same time, the platform provides the flexibility for customization—developers can, for example, set up explicit port bindings for containers, co-locate them, and configure routes and dependencies. For flexibility in managing and automating containers, PKS exposes the Kubernetes API.

DevOps or a platform operations team is likely to play a key role in managing PKS and in providing a system and tools for continuous delivery, such as Jenkins, a pipeline automation tool.

PKS integrates with the Pivotal Cloud Foundry Marketplace, a service catalog that give developers instant access to a library of add-on services, such as databases, tools, identity management, and messaging. The services in the catalog include MongoDB, Elasticsearch, MySQL, Redis, Jenkins, PagerDuty, Single Sign-On, RabbitMQ, and Kafka.

The ideal use cases and workloads for PKS are as follows:

- Repackaging applications in containers when they require customization, fine-grained control, or orchestration.
- Replatforming traditional applications so they can be orchestrated with Kubernetes for high availability and scalability.
- Running modern data services such as Elasticsearch, Cassandra, and Spark.
- Running ISV applications packaged in containers.
- Running microservices-based apps that require a custom stack.

Pivotal Cloud Foundry Use Cases

Pivotal Cloud Foundry dictates that you use its prescribed approach for scheduling and running containers as well as its automated system for CI/CD.

Developers interact with application code. They commit and push their code, and then the platform builds the application, creates its supporting environment, and launches the app. The platform provides container images, automates the building of containers, and implements container networking. The platform also handles quotas, logs, monitoring, usage tracking, and monitoring. A built-in marketplace gives developers access to backing services, such as databases, so they can easily set them up as microservices.

PCF fits situations in which developers want to focus on writing an app and do not care how it is deployed. It is an ideal platform for building 12-factor apps, especially when you do not need the flexibility to use standard container APIs. PCF is commonly used to build applications with Spring, NodeJS, .NET, and Java.

Pivotal Cloud Foudry can be combined with PKS so that developers can consume Kubernetes as a service. For more information on Pivotal Cloud Foundry, see Developer-Ready Infrastructure.

Conclusion

Different container platform are optimized for different use cases, workloads, levels of developer engagement, and do-it-yourself customization. vSphere Integrated Containers best serves use cases in which an app is to run as an isolated container instance. VMware PKS provides a Kubernetes-based container services that gives you the flexibility to implement a custom stack for building and deploying applications. Pivotal Cloud Foundry supplies a complete, ready-to-use application platform that requires no do-it-yourself customization but also provides no access to standard container APIs.

VMWARE PKS AT-A-GLANCE

VMware PKS provides a highly available, production-grade Kubernetes-based container service equipped with advanced networking from VMware NSX, a secure image registry called Harbor, and lifecycle management with BOSH. The solution radically simplifies the deployment and operation of Kubernetes clusters so that you can run, manage, secure, and maintain containers at scale on VMware vSphere.

KEY BENEFITS OF VMWARE PKS

- Quickly provision Kubernetes clusters on-demand
- Deliver high availability for Kubernetes components with rolling upgrades, health checks, and auto-healing
- Use advanced container networking with micro-segmentation, load balancing, and security policies
- Secure container images with vulnerability scanning and image signing
- Improve operational efficiency with monitoring, logging, and analytics

LEARN MORE ABOUT CLOUD-NATIVE SOLUTIONS FROM VMWARE

To find out more about how VMware can help you build, run, and manage cloud-native applications, see https://cloud.vmware.com/cloud-native-apps.