E-guide

PRO+ Content

Your Guide to Container Management

In this e-guide

In this e-guide:

Even if containers are not currently deployed in your organization, with the rate at which they are currently gaining speed in the IT world, it's likely you will evaluate them for your enterprise apps at some point in the near future.

Download this guide – either for your current container strategy or for future reference – to help you in working through the management of containers, deciding on a containers-as-a-service provider, and how to deploy containers in production.

In this e-guide

To manage and monitor containers, understand their peculiarities

Phil Sweeney, Senior Managing Editor

http://searchitoperations.techtarget.com/ehandbook/A-fresh-approach-is-needed-to-manage-containers-properly

A good IT operations team is primed to worry about an application's availability and performance. It's a fundamental concern in every hour of every day. So when containers begin to deliver apps, an IT team needs to implement new methods to monitor containers -- and to adjust to new ways of worrying.

Container technology brings advantages to an IT infrastructure, particularly with improved use of resources. The value of containerization, at least from a technology point of view, is mostly clear by now. What holds back adoption is uncertainty over how to work with and monitor containers. They come in different formats. They are built through different methods. They behave in different ways. They require a management approach that is, yes, different.

Momentum behind containerization is building. A recent report from 451 Research pegged spending in the application container segment at $762 million in 2016; by 2020, that figure is projected to jump to $2.7 billion.

That kind of growth suggests containers are destined to find a place in plenty of IT shops, which makes learning about them -- and how to manage them -- an IT priority. This handbook tackles that mission, beginning with an overview from IT consultant Alastair Cooke on the best ways to manage and monitor containers -- either through native tools or emerging third-party products. There will be plenty of new tasks for IT admins to consider in a container environment. Also included are discussions of how Docker is adjusting its management tools and how container security is evolving as adoption begins to pick up.

Next article

In this e-guide

What admins need to know to master containerization technology

Alastair Cooke,

http://searchitoperations.techtarget.com/feature/What-admins-need-to-know-to-master-containerization-technology

Containers have inserted themselves into the IT conversation, and their usefulness is being discussed in ever-widening circles. While adoption has been limited so far, it seems clear that 2017 is going to see a lot more production deployment of applications in containers. Deploying anything into production, of course, is when the operations team gets involved -- and they're going to have questions.

There are plenty of issues to consider when it comes to management of containerization technology. These include how to handle dependencies in files and the role microservices may or may not play in your cloud strategy. To be sure, there's a learning curve.

DevOps brought the idea that developers should support production. The reality is that developers need their sleep, and it's the operations team that looks after production at all hours of the day and night. Operations teams will need to understand the impact of these new containerized applications, and they are the ones expected to resolve availability or performance issues.

Also, an IT organization might need new tools to monitor and manage containers. Containers, based on how they are used, can have important operational impacts on an IT organization. To properly manage containerization technology means admins must take these factors into consideration and figure out a plan to make it all work.

Manage applications and dependencies

Sometimes containers are just a way to package and distribute an application. When an existing application becomes distributed as a Docker container, for example, that packaging is essentially the special value that Docker provides -- although it's not the only reason to use Docker or containers. Containers existed before Docker, which is used to create containers in a structured and controlled way.

In some cases, the container simply wraps up the application and its dependencies. The container is then run on a server. The magic of Docker is to wrap all of the application's dependencies into a Docker image and have a single text file (Docker file) that describes how to create the image. With this existing application model, each server may run just one instance of the container -- just like the server used to run one instance of the application.

It's fairly simple to monitor and manage this use of containers: continue to monitor and manage the server. The admin can still see the application processes on the server, along with resource usage.

In some ways, this is even simpler as there is no need to check that the server has all of the application's prerequisites. These prerequisites include the correct version of Java or the appropriate Python libraries. All of these dependencies are in the Docker image and controlled by the Docker file.

It might not be necessary to install and maintain Java on the server. But you should have some control of these Docker files and ensure that the Docker images contain components that are up to date. Rather than update or patch the Java on the server, you need to update the Docker file and build a new Docker image.

One new operational task may be to scan the Docker files for vulnerabilities. A Docker file may direct the installation of vulnerable or unsupported components into the image. It may also be necessary to implement policies about the maximum allowed age of a built Docker image. The versions of the dependencies are fixed inside the image and can only be updated by building a new image.

Containerization technology is dynamic

Sometimes containers enable microservices architecture. This makes them a whole new way to assemble applications.

Containers can be started and stopped much faster than VMs. Starting another container typically takes a fraction of a second. A single server can run multiple containers at once, with some isolation between the containers. In applications built in the microservices way, containers will be far more numerous and often short-lived. The application is broken into small parts -- microservices -- and will have a Docker image for each of the dozens of small parts. Each microservice can scale in and out by creating or destroying containers.

This is a far more dynamic environment than the monolithic applications that admins are used to managing. The underlying servers can still be monitored with your normal tools, but the containers themselves are too volatile for these tools.

A single container may only live for a few seconds. New tools will be required to manage and monitor these microservices applications in production. Hyperscale vendors such as AWS have built their own tools to manage their fleets of containers. Large organizations are more likely to use tools from providers such as New Relic and Datadog to monitor their container fleets.

Schedulers, tools make a difference

An essential part of DevOps is to have automation around all parts of the application, which includes the many containers. This is where schedulers play their part: to make sure that the right number of each container is running and healthy. The scheduler may be something container-specific such as Kubernetes or a more general-purpose scheduler such as Apache Mesos. It might even be Kubernetes and Mesos together.

The automation reduces the amount of manual intervention required to monitor and manage the application. Management should be based on setting policies for the application. The scheduler simply implements the policies.

To deploy monolithic applications into production with containerization technology may be a small change to the management and monitoring of those applications. If the applications are broken up into microservices and require a scheduler, then there will be far more significant operational changes.

While it is still early in the development of the operational tools for fleets of containers, there are a lot of tools being built for container management. Successful operations of microservices applications will need new tools and methods for operations teams.

Next article

In this e-guide

Deploy containers in production with clustering

Mike Pfeiffer, Chief Technologist

http://searchitoperations.techtarget.com/tip/Deploy-containers-in-production-with-clustering

Deploying one or two containers on a laptop during development is easy; building a production-grade application without any additional tooling is a different story.

Most containerized applications in production comprise multiple containers spread across a number of physical and/or virtual hosts. Complexity increases with automated container deployments, and when IT shops must scale containerized applications to accommodate additional demand.

To deploy containers in production, IT teams must find a way to automate, scale and manage complex architectures. Container orchestration tools make life easier.

In this e-guide

Container orchestration and clustering capabilities

Live applications need support for load balancing, rolling updates, monitoring and auto scaling. Container orchestration tools like the open source Google Kubernetes bridge the gap from development to production.

Container orchestration tools manage complex applications comprised of multiple containers running on a cluster of host machines. IT teams deploy containers to physical host machines or virtual machines as an additional abstraction layer, running on hardware on premises or in the public cloud.

There's a learning curve when it comes to implementing container orchestration tools. The architecture can initially seem complex. But production-grade containerized applications require a capable management system to deploy containers. Making an investment in orchestration and clustering technology, such as Kubernetes, can set up a team with the agility to move from on-premises data centers to the cloud.

Kubernetes is considered one of the most mature container orchestration tools, with the most features for enterprise-level container deployments. Kubernetes is made up of several building block components:

- Controllers are systems used to manage a container cluster. They ensure that all of the systems deployed within the cluster are configured properly and adhere to a desired state.

- Nodes, also referred to as minions, are the physical or virtual machines running the Docker Engine for containers. They can host various containers running within a managed cluster. A master controller manages nodes, exchanging messages using ETCD, a key/value management and communication service.
- Pods are a collection of one or more containers deployed on a node. When a pod is comprised of multiple containers, the cluster controller guarantees that each of those containers resides on the same node. Containers within each pod share resources, like disk volumes, if needed. Each pod is assigned a unique IP address within the cluster, which eliminates port conflict issues with other pods running on the same node.
- Labels and selectors identify objects in the system. Labels are simply key/value pairs that identify the components within the system. Once the administrator sets up objects with labels, they can use selectors to query all of the objects that share the same label. They can also take actions on a group of components sharing the same label. For example, using single command, a developer can shut down all of the containers labeled with a particular version number.
- Services are a critical piece of the container-deployment puzzle. Services are endpoints that enable multiple pods running across different nodes to work together to power a complete application; the pods within a service are defined using label selectors. Because all containers within a pod are guaranteed to run on the same node, availability increases when the application uses multiple pods distributed across multiple underlying hosts. Services provide built-in load balancing for container deployments so that requests are distributed across each of the pods in a round-robin fashion.

Key benefits to running Kubernetes

Kubernetes was initially developed by and run inside Google for several years before it went open source. The container cluster manager is backed by a large community of developers that moves the project forward on a regular basis. You can run Kubernetes anywhere without worrying about investing in a proprietary technology that may be specific to an on-premises deployment or public cloud platform.

Teams can run Kubernetes on premises, on physical servers or virtual servers, and in a hybrid deployment. This also creates the opportunity to run it in hybrid deployments. And because it runs on VMs, Kubernetes deploys containers on servers running within Amazon Web Services, Microsoft Azure, Google Compute Engine or any other infrastructure as a service cloud provider.

Next article

Unwrap containers-as-a-service challenges and best practices

David Linthicum, Cloud analyst

http://searchcloudcomputing.techtarget.com/tip/Unwrap-containers-as-a-service-challenges-and-best-practices

Most enterprises today believe containers will drive value, and, in many cases, those organizations want to use containers through a cloud service -- a model known as containers as a service -- rather than on premises.

Here's a breakdown of containers as a service benefits, challenges and current options on the market.

Containers-as-a-service benefits

The reason enterprises want to use a cloud-based container service, rather than deploying containers on premises, is similar to why they want to use compute servers and storage systems in an infrastructure-as-a-service cloud. These reasons include:

- More efficient sharing of resources;
- Smaller or no hardware and software footprint on premises;
- Huge cost advantage; and
- Centralization of repositories for container image distribution.

With containers as a service, organizations can avoid the complexity and cost of setting up their own container development, deployment and runtime environment by renting a fully equipped environment from a cloud provider. Since it is a service, the cloud provider makes any updates to tools, images and repositories on their users' behalf. In addition, providers typically have an ecosystem built around their containers-as-a-service offering, allowing users to obtain databases, security, management and other services to support their container deployment.

Containers-as-a-service drawbacks and challenges

Containers as a service is not a perfect model; it comes with its own brand of challenges. For the most part, the biggest challenge when dealing with cloud providers is integration with external resources, meaning resources that reside on premises or on other cloud platforms. This integration doesn't come naturally for most containers-as-a-service offerings, since they typically require custom configurations or redevelopment to enable communication with these external resources.

Other issues include container-based security, such as encryption services, which can also be difficult to integrate, since users might need to build an API call from the containers to those services. There are third-party tools that can help with these container integration challenges, such as Aqua Security and Twistlock, but organizations should carefully map out their requirements before choosing one.

Containers-as-a-service best practices

Before picking a containers-as-a-service offering, understand your core requirements, including those for performance, security, governance and management. When it comes to choosing a containers-as-a-service provider, enterprises sometimes get distracted by big brand names, such as Amazon Web Services (AWS), and ignore the core issues they are attempting to solve.

If you overlook compliance issues, for instance, they could toss a monkey wrench into your containers-as-a-service deployment, potentially forcing you to move back on premises or to evaluate another containers-as-a-service provider. Check if your containers-as-a-service provider has compliance features that can help you solve these problems.

Learn all you can before moving to a containers-as-a-service offering. This includes taking time to understand what you need, when you need it and creating a container vision for your enterprise.

Options in the market

So, which containers-as-a-service offerings can you choose from today? Azure, AWS and Google all have services that support Docker containers. These options are Azure Container Service, AWS Elastic Compute Cloud (EC2) Container Service (ECS) and Google Container Engine (GKE).

Azure Container Service, the newest of the three services, is based on Apache Mesos. This open source container orchestration system is often preferred by those who are moving from a Windows environment.

Mesos, which is used by companies such as Netflix and eBay, is considered one of the most scalable container orchestration services. Mesos competes with Kubernetes, Google's open source container management system, as well as Docker Swarm.

Last year, AWS ECS had a number of operational issues, such as the inability to monitor containers at a fine-grained level. AWS has since fixed that issue, as well as other issues around performance. Out of all the major vendors, AWS' containers-as-a-service offering seems to be the most improved compared to last year.

GKE remains a strong player, with tight integration for AWS CloudWatch, as well as Kubernetes. GKE also provides updates that align closely with the Kubernetes release cycle -- a helpful feature for enterprises, since Kubernetes clusters can be a challenge to deploy and upgrade.

Before choosing a provider, check your enterprise requirements and compare them closely to what each provider can offer. Also, consider your future requirements, and make sure the container service will continue to grow and meet those needs.