MANAGING CONTAINERS AND CODE FOR DEVOPS

Containers help applications run reliably in a number of different environments, from testing to deployment, on a variety of devices, both large and small. Using containers in the development process creates a solution to the problem of how code runs in non-identical environments. In short, by encapsulating an application, its dependencies, libraries and other elements required to create a full-runtime environment, containers offer a form of lite virtualization that is identical across environments.

Here's what you need to know about containers and code in your DevOps environment:

- What are containers?
- What are the benefits of using them?
- What does the future hold for containers and code?
- How can you implement containers?
- What are some considerations when making the transition to containers in your coding environment?

Stay tuned to learn all about containers and their relationship with code in this comprehensive article.

(This article is part of our DevOps Guide. Use the right-hand menu to navigate.)

What Are Containers and Why Should You Care?

Containerization is a lightweight form of virtualization. Unlike a virtual machine which uses a hypervisor with its own kernel and therefore offering a heavy-duty suite of resource-intensive

computer processes, each container accesses a single OS kernel. Multiple containers can exist side-by-side, each containing an environment equipped to handle an application, its dependencies, libraries and other necessities using fewer resources.

Docker was the first brand of containerization to hit the market in 2013, and since then other big names in technology have created co-existing engines and competing products, such as Google Kubernetes engine. As container applications become more popular for their lightweight agility and ease of use, here's what you need to know about containers for beginners:

- **Containers are faster:** They are also leaner and more reliable than other forms of virtualization.
- **Containers are versatile:** They run on Linux and Windows and across a number of hardware specifications.
- **Not cost prohibitive:** As far as virtualization technology is concerned, containers are relatively inexpensive to implement.
- **Ease-of-use:** Containers help simplify modern application development.
- **Flexible and location agnostic:** Perhaps the biggest benefit is containers create identical environments regardless of location.

Due to its many benefits, containerization is on the rise. For companies focused on agile development practices, this may lead to new questions and challenges around managing code and containers. In the following sections, we will examine containers and code in common scenarios that you may experience.

How to Implement Containers in Your Organization

There are a few common trains of thought on how to best implement containers. Depending what expert you ask, you may get a different answer. We recommend changing over to containers, incrementally, beginning with the build environment in a CI/CD framework. Below is a broad spectrum DevOps approach on how to handle this transition:

Consider Requirements:

Install the desired engine on server or servers. Find a service that runs registries. You'll need a registry that allows you to select from container images. Alternatively, you may run your own.

Containerize Your Build Environment:

Create a new file based on your chosen container image. For those not familiar with the language, an image is the standard specification used to build a new container. If you use a configuration management tool already, you'll want to choose a tool you know is compatible. A Dockerfile, for example, should work within existing configuration management processes.

In Docker, you can build an image using Docker build function and push it to the Docker registry, which is the library of images that can be accessed when you need it. After creating a new Dockerfile based on your chosen image, continue to build the application with all of its dependencies and libraries.

Next, you'll need to change the existing CI build scripts to push the new application image to the Docker registry.

Containerize the Deploy Process:

Hopefully, your CD is already compatible with Docker, as this will simplify the process. You can deploy directly to Docker or deploy the file in another engine environment like Kubernetes. Whichever you choose, you'll want to make sure to deploy other tools like those for management and monitoring.

Infrastructure as Code

The concept of Infrastructure as Code (IaC) is a relatively modern one but is already having a profound impact on DevOps. The idea is to automate a declarative framework that focuses on what applications must be built, not how they work. That's what sets IaC apart from configuration management resources. Experts like Puppet Labs have been pioneering IaC technology, with a platform that looks for issues in coding and automates to bring the issues into the correct state.

IaC was made to complement DevOps by automating processes that continue to extend the reach of where the application ends and development environment begins. The resulting ambiguity is a common theme in the best tried and true DevOps practices.

Ultimately, containerization and IaC should complement each other more than collide, and that's an integration that developers will continue to strengthen as DevOps continues to evolve into the enterprise norm, the gold standard.

Other Considerations: Containers and Code

When considering a transition to containers and DevOps, keep these best practices in mind:

- **Seek out seamless integration with existing IT infrastructure:** This is a primary draw of containers.
- **Consider security and compliance:** Integrating containers in this environment is a challenge in the current state of DevOps.
- **Centers IT pros:** By making IT an instrumentally part of company-wide operations.
- **Supportive of hybrid IT environments:** Data shows enterprises are running containers on other VMs and using FaaS, too.
- **Great for fans of Kubernetes:** With Kubernetes as the gold standard in framework orchestration, containers make a lot of sense.
- **Supports transition to microservices and DevOps approach:** The addition of microservices to the development cycle allows developers to rapidly deploy new features without a full release.
- **Container management is application-centric.**
- **Transition to declarative management:** Declarative management is where developers define the desired state and automate to that goal.
- **Can be used with other services**: Other services include PaaS and FaaS.

For these reasons and more, it's beneficial for enterprise businesses to make a priority of containerization. They are agile, scalable, and affordable, making them a great option for any business looking to further support a DevOps approach.

Containers and Code for DevOps

As development continues to evolve with DevOps at the forefront of efficiency and business agility, developers must come to terms with the idea that containerization is here to stay. Containers provide a solution for the commonplace DevOps challenge of getting an application to perform the same way in all environments. They do this using a slice of the resources required to run a virtual machine, meaning dozens of containers can run as economically as one VM. They offer speed, flexibility, and ease of use.

The challenge will become how to get containers to interact with other services, like the ever expansive infrastructure as code (IaC). We believe containers make IaC easier, by reducing the confusion caused when troubleshooting infrastructure as code in multiple environments.

Tools of the Trade

There are several tools available right now to assist in containerizing IaC, among the experts throwing their hat in the ring are Chef, Ansible, Terraform, Puppet, and SaltStack. Today, BMC offers a tool to support all of your digital business automation, including:

- Accelerated service delivery
- Change response and management
- Faster application development and feature releases
- Supportive of DevOps goal to bridge development and operations
- Automates and manages workloads
- Workflow orchestration
- Eliminates challenges of data management and availability, and more