5 Key Benefits of Docker: CI, Version Control, Portability, Isolation and Security

dzone.com/articles/5-key-benefits-docker-ci

April 29, 2015

Docker doesn't need an introduction. It is one of the hottest open source projects that allows you to deploy your application inside containers, adding a layer of abstraction. In a seemingly constant state of maturation, the benefits of using Docker increase on a regular basis. In this post, instead of talking about what Docker is or how it works, I'll outline the top five benefits of using the ever-growing platform.

Continuous Deployment and Testing

Docker is gaining a lot of traction in the development and devops world for its consistency across environments. There are always minor differences between environments in development and release lifecycles, unless you have your own private repository environment with tight checks in place. These differences may be because of different package versions or dependencies. Nevertheless, Docker can address that gap by ensuring consistent environments from development to production. Docker containers are configured to maintain all configurations and dependencies internally. As a result, you can use the same container from development to production making sure there are no discrepancies or manual intervention.

With Docker containers, you can also ensure that developers don't need an identical production environment set up. Instead, they can use their own system to run Docker containers on VirtualBox. The beauty of Docker is that you can run the same container on Amazon EC2 instances. If you need to perform an upgrade during a product's release cycle, you can easily make the necessary changes to Docker containers, test them, and implement the same changes to your existing containers. This sort of flexibility is a key advantage of using Docker. Just like standard deployment and integration processes, Docker allows you to build, test and release images that can be deployed across multiple servers. Even if a new security patch is available, the process remains the same. You can apply the patch, test it and release it to production.

Multi-Cloud Platforms

One of Docker's greatest benefits is portability. Over last few years, all major cloud computing providers, including Amazon Web Services (AWS) and Google Compute Platform (GCP), have embraced Docker's availability and added individual support. Docker containers can be run inside an Amazon EC2 instance, Google Compute Engine instance, Rackspace server or VirtualBox, provided that the host OS supports Docker. If this is the case, a container running on an Amazon EC2 instance can easily be ported between environments, say to VirtualBox, achieving similar consistency and functionality. This grants you a level of abstraction from your infrastructure layer. In addition to AWS and GCP,

Docker works very well with various other IaaS providers like Microsoft Azure, and OpenStack, and can be used with various configuration managers like Chef, Puppet, and Ansible, to name a few.

Environment Standardization and Version Control

As discussed above, Docker containers ensure consistency across multiple development and release cycles, standardizing your environment. On top of that, Docker containers work just like GIT repositories, allowing you to commit changes to your Docker images and version control them. Suppose you perform a component upgrade that breaks your whole environment. It is very easy to rollback to a previous version of your Docker image. This whole process can be tested in a few minutes. When compared to VM backup and image creation processes, Docker is fast, allowing you to quickly make replications and achieve redundancy. Additionally, launching Docker images is as fast as running a machine process.

Isolation

Docker ensures your applications and resources are isolated and segregated. A few months back, Gartner published a report stating Docker containers are as good as VM hypervisors when it comes to isolating resources, but there is still work to be done in terms of management and administration.

Consider a scenario where you are running multiple applications on your VM. These applications can be team collaboration software (e.g., Confluence), issue tracking software (e.g., JIRA), centralized identity management systems (e.g., Crowd) and so on. Seeing as all of these applications run on different ports, you would have to leverage them on Apache and Nginx as a reverse proxy. So far, everything is in good shape, but as your environment moves forward, you will also need to configure a content management system (e.g., Alfresco) into your existing environment. Bear in mind that it requires a different version of Apache Tomcat, which will cause a problem. In order to fix this, you can either move your existing applications to another version of Tomcat or run your content management system (Alfresco) on your currently deployed version.

Fortunately, with Docker, you don’t have to do this. Docker makes sure each container has its own resources that are isolated from other containers. You can have various containers for separate applications running completely different stacks. Aside from this, effectively removing applications from your server is quite difficult and may cause conflicts with dependencies. However, Docker helps you ensure clean app removal since each application runs on its own container. If you no longer need an application, you can simply delete its container. It won’t leave any temporary or configuration files on your host OS.

On top of these benefits, Docker also ensures that each application only uses resources (CPU, memory and disk space) that have been assigned to them. A particular application won’t hog all of your available resources, which would normally lead to performance degradation or complete downtime for other applications.

Security

Docker is evolving at a fast pace, which Gartner even acknowledges, as mentioned above. From a security standpoint, Docker ensures that applications that are running on containers are completely segregated and isolated from each other, granting you complete control over traffic flow and management. No Docker container can look into processes running inside another container. From an architectural standpoint, each container gets its own set of resources ranging from processing to network stacks.

As a means of tightening security, Docker uses host OS sensitive mount points (e.g., '/proc' and '/sys') as read-only mount points and uses a copy-on-write filesystem to make sure containers can't read each other's data. It also limits system calls to your host OS and works well with SELinux and AppArmor. Additionally, Docker images that are available on Docker Hub are digitally signed to ensure authenticity. Since Docker containers are isolated and resources are limited, even if one of your applications is hacked, it won't affect applications that are running on other Docker containers.

Final Note

In conjunction with cloud computing, the benefits mentioned above definitely demonstrate how Docker is an effective open source platform. While the list of benefits could go on, I simply wanted to highlight the top five for you today. If you use Docker, feel free to share your use cases or any benefits you've experienced.