Abstract

There are often many roadblocks that stand in the way of easily moving your application through the development cycle and eventually into production. Besides the actual work of developing your application to respond appropriately in each environment, you may also face issues with tracking down dependencies, scaling your application, and updating individual components without affecting the entire application.

Docker containerization and service-oriented design attempts to solve many of these problems. Applications can be broken up into manageable, functional components, packaged individually with all of their dependencies, and deployed on irregular architecture easily. Scaling and updating components is also simplified.

Containers are meant to be completely standardized. This means that the container connects to the host and to anything outside of the container using defined interfaces. A containerized application should not rely on or be concerned with details about the underlying host's resources or architecture. This simplifies development assumptions about the operating environment.

*My mission is to containerize the existing environment and automate the process of building, testing and deploying, and hence enhance the process of Continuous Integration and Continuous Delivery (CI/CD) using DevOps tools (**On-premise**).*

Benefits of the new container system over the existing system:

- ✧ Easy Scalability
- ✧ Simple Dependency Management and Application Versioning
- ✧ Extremely lightweight, isolated execution environments
- ✧ Shared Layering
- ✧ Composability and Predictability