Containers "Cheat Sheet"

Denise Boehm

Note: In 2020, I created this informal backgrounder as a precursor to developing positioning, messaging, and GTM plan for app hosting with containers/Docker, and Kubernetes.

Docker – One of Several Container Options That Is Widely Used

From Wikipedia – **Docker** is a set of Platform as a Service (PaaS) products that uses OS-Level virtualization to deliver software in packages called containers.

So, in **plain words**:

- Docker is technology developers use to create an application in a container.
- The container holds everything you need to run the app. It has no external dependencies, containing the app, the OS, relevant libraries, and any other components the app needs to function.
- *Business Value*: Cost and time-efficient applications speed time to market.
  With containers, you are released from a complex application development platform with numerous dependencies on constantly evolving technologies, such as libraries, databases, and OSes. Containers allow DevOps to deliver applications and their components as loosely coupled, independent components that can be maintained and updated separately without breaking the application.

Example

Scenario: You've containerized your app, tested it, deployed it, and it's humming away. But wait, what's that? A new version of Red Hat offers more features in the OS that better support your app. However, both the CEO and the developer are concerned about possible time-consuming dependency issues with the app and its components, e.g., the database or networking. Testing could take a while and have a negative impact on delivering business value to customers.

Customer Value: Instead of upgrading the OS, then testing, and fixing every component of the application, you leave your app in production in the current container, then:

- DevOps creates a new container with all the same components with the OS upgrade.
- You can quickly swap out the original container with the old OS for the new container with the updated OS, with these benefits:
  - No downtime.
  - Less effort for application changes because developers can modify individual app components instead of taking on the application and its dependencies as a monolith.
  - Better functionality, faster deployment, and seamless upgrades.
  - *Most importantly*: Satisfy customers with swift time to market.

Microservices – The How of Containers

- Microservices is the name for the separate, independent components in the container that are mentioned above.

- These are the parts of an application environment that make your application run the way you want it to.
- In the container, each “microservice” is decoupled from the other services in the container. That’s why you can make changes to one thing and swap out the containers so easily.
- You aren’t recreating the wheel every time you make an app upgrade. Just swapping the old container out with the new one that has additional/modified/updated components.

How do those microservices know what the others in the container are doing and how to interact with them?

- The short answer is by using Application Programming Interfaces (APIs):
    - Microservices use APIs to communicate with each other in the container.
    - APIs can also reveal application data and functionality to third parties for solution integration.
    - Many APIs are “commoditized” and public, so-called Open APIs.
        - For example, Google and other software solutions create free APIs that an app can interface with, so adding Google functionality to external offerings. Win for Google and user.
        - Open APIs are the secret sauce for creating SaaS solutions. For example, a retailer could create a SaaS offering that incorporates external software functionality through an Open API. You don’t have to recreate what already exists.
- Other API facts:
    - APIs usually send data by means of HTTP requests.
    - HTTP requests usually receive a textual (human readable) response.
    - The format those requests come in is usually JSON, a data-interchange format that makes it easy for humans to read the API and write to it.
    - JSON is also easy for machines to parse and generate.
    - JSON stands for JavaScript Object Notation – no one will ask you that, but they may use JSON in their questions to you about microservices and APIs.