Abstract

Containers have become an indispensable part of public IT systems. From huge enterprise level demands to hobbyists, the use of containers improves performance of systems while allowing for highly scalable infrastructures able to meet virtually any demands. Their use in military applications, however, is just beginning. While “containers” may go by different names and have slightly different nomenclatures and operational processes, for practical purposes they all function very similarly. Docker and Kubernetes are probably the most well-known and are virtually ubiquitous when speaking about containers.

Containers are configured to run processes based on events or as scheduled by a container orchestrator such as IronWorker, which we will discuss more later. For example event A triggers Container cluster W to carry out process Y. The beauty of containers is that once configured, they automatically scale depending on the processing needs. In a traditional virtualized server, these demands quickly create bottlenecks as tasks are queued and processed synchronously. In a containerized environment the demands are spread across the necessary quantity of containers. Each container processes tasks asynchronously, or in layman's terms, all at once without eating away at operating system resources, which both reduces performance and increases complexity compared to containers.

If we make this example real-world it is easy to understand. All modern militaries utilize some of the same overarching theories that are the basis for containers: interoperability, compartmentalization and standardization. Containers allow the decoupling of processes from operational systems while software such as IronWorker abstract the control of the processes and scale automatically with built in redundancy. Beyond that, they offer unprecedented flexibility in deploying to different environments (cloud, on-premise / closed network, different servers, etc.)

In short, a containerized environment creates a faster, more robust data processing capability, while reducing system workload demands.

IronWorker further improves on this by orchestrating and managing containers. This is now frequently called “serverless.” While there is no truly “serverless” system, because the operation and function of what previously relied on human configuration is abstracted and performed by the platform, it is called serverless.

Serverless platforms such as IronWorker create a microservice which is efficient and self-contained. Because IronWorker is language and environment agnostic, seamless integration across virtually any scenario is fast and easy. This becomes increasingly more important each day as battlefields and the world in general become more complex. IronWorker is the only tool on the market with these capabilities.

Potential use cases

The implementation of containers can cover nearly any application. As such, we will discuss specific possible uses. However, these are not meant to be an exhaustive exploration. In fact, we will only scratch the surface.

1) On-premise / stand-alone applications.

If we think about semi-closed systems such as aircraft, the use of IronWorker to improve system performance and efficiency becomes immediately apparent. The improvement in processing speed and robustness, while reducing overall demands on the aircraft's data processing systems, translate directly to mission success. As the quantity of data acquired by the aircraft increases with technologies such as sensor fusion, so too have technical difficulties in processing that data into useful information. Using containers alleviates many of the potential issues with processing huge amounts of data. Not only that, but using a tool such as IronWorker within individual aircraft allows for the scaling of system resources during times with more, or less, sensor information being acquired. This could have potentially profound effect on the performance of creating real-time integrated battlefield views across a virtually unlimited number of endpoints.

The same is true for virtually any battlefield equipment. From tanks to infantry transmitting location data, communications, etc., containers run by IronWorker in stand-alone deployments improve data transmission.

2) Satellite image and sensor data processing

With every improvement in satellite technology, larger and more complex data sets are created. IronWorker has already been proven as a useful tool in processing not only satellite data, but image processing as well. It has been used to turn days of processing into minutes. Because it's scalable, when no data needs to be processed, the system load is reduced to virtually nothing. When it needs to spin up again, the process is fast and reliable.

3) Spatial data processing and telemetry

Radars, satellites, sonars, and other sensors all create large quantities of data that is must be processed with as little latency as possible. This creates huge computing demands. IronWorker reduces that demand, which can free up system resources and increases reliability. Because of that, data can also be processed more effectively at each terminal, then sent to other terminals where it can be used to create a more complete and better integrated spatial picture.

4) Closed / classified network use

While this is somewhat implied in the above use-cases, it is important to note that IronWorker can be deployed and used on any existing infrastructure, even those which do not have an outside connection or link. IronWorker is the only way to run serverless container orchestration

without utilizing a public or private cloud. Especially in the cases of testing and evaluation of classified systems, where data processing must be done in-house, a tool like IronWorker can speed up testing cycles by magnitudes. IronWorker is also easily migrated anywhere, including from on-premise to any cloud. One simply saves their configurations and uploads them to the new environment.

Conclusion.

IronWorker can serve as an invaluable tool across thousands of applications. These range from strategic to tactical.