Containers IO: Does It Have to Be a Painful Coming of Age?

The notion that IT operations (IO) must develop a business acumen is not new. Lately though, with technologies such as containers, enterprises are experiencing a renewed urgency to accept and implement the idea. Often this is preparatory for coming shifts in the IT landscape (such as IoT or Big Data), as well as to overcome problems such as:

- Inability to scale
- Continual downtimes
- Deployment delays
- Inefficient build pipelines
- Resource utilization issues
- Growing cloud waste and cost

As recent analyst reports indicate, containers will soon be ubiquitous, and “mature” will be an apt way to describe them. What’s often not fully grasped, though, is the current development stage of the containers infrastructure, especially from the enterprise viewpoint: adolescent.

The extent to which containers operationally affect infrastructure is not yet widely understood. Enterprises must look closely before they leap into containers. In terms of IO, containers introduce a magnitude of complexity heretofore unseen. This is due to the fundamental differences between a containers infrastructure and a traditional infrastructure, which boil down to the following for containers:

- Much higher density of services
- Faster deployment pace for new features
- Container-enabled developer innovation using different software stacks, without overburdening devops
- Increased attack surface area for a malicious hacker (every single container, and the software inside, is an attack vector)
- Increased complexity of enforcing organization-wide policies and controls

Not understanding the above will only lead an enterprise into the worst kind of cluster: a poorly planned containers infrastructure that has major growing pains–some of which will be significantly more problematic than those the containers were meant to solve.

Operating Containers: A New Level of Complexity

An infrastructure that does not meet the operational requirements for containers will experience serious issues, including:

- Infrastructure interruptions
- Ineffective Monitoring (more “noise” than “signal”)
- Lack of holistic view of infrastructure
- Poor Quality of Service of the infrastructure as a whole
- Undetected anomalies
- Ineffective security - traditional security will fail with containers
- Inefficient intelligence from business logic layers
- Data-access authorization issues for unstructured data (or Big Data)

Additionally, the original set of problems may still not even be solved for. For enterprises in this quagmire, the problem is due to a poor understanding of the complexity of container operations.

Containerizing an Infrastructure

> ***Containerization increases the operational complexity of an infrastructure by a magnitude that’s much higher than monolithic applications or components.***

That operational complexity also applies to all requisite infrastructure services (all of which must be scalable). These include container orchestration, servicing an application’s development life cycle, and providing the mission-critical services necessary for containers in production (briefly described below).

1. Orchestration: As the number of containers and hosts grow, automating container deployment (configuration, ongoing monitoring, troubleshooting, rolling upgrades, rollback) is critical to scaling an application.

2. SDLC via CI/CD: Whether an enterprise is planning to implement CI/CD to enable containers, or whether it will be translating current CI/CD to do so, proper infrastructure support is vital. Though containers are conducive to CI/CD (for example, modifying and updating code is easier, as is concurrent releases by multiple teams), they do not handle all CI/CD requirements.

When containerizing an application development life cycle, you must attend to composition, scheduling, networking and storage. Also, you must:

- Build an application or service with multiple interconnected containers
- Be able to schedule them to run across multiple hosts
- Plan for container communication across multiple host systems
- Provide the necessary testing framework–both functional and nonfunctional
- Ensure application or service is secure and not vulnerable

3. Mission-Critical Services: In production, containers have specific requirements for the necessary services (security, performance and availability management and monitoring). These include:

Persistence: The phrase, stateless applications, is misleading. State is needed, just somewhere else. That state, and the mode of transport there, are vital. If the data store takes a long time to start, the benefit of containers is lost. Applications need storage provisioned at runtime (think data in motion, significant amounts of which must sometimes be quickly processed in real time).

Security: Traditional monitoring tools won't cut it in a containers architecture. Though perimeter security is needed, it's not adequate. The distributed-systems nature of containers architectures effectively dissolves the perimeter (a majority of attacks are launched from the inside). A zero-trust model is key given the lack of control over lateral movement and zero visibility as to where things are moving. To be effective, security must be holistic, workload-specific and adaptive.

Monitoring: The meaning of tracked metrics must be identified immediately. This is no small task given the purposeful fragmentation of a container ecosystem. How containers have changed monitoring is due to:

1. Their ephemeral nature
2. Proliferation of objects, services and metrics to track
3. New methods of monitoring
4. New focal points to monitor (services)
5. Expanded group of monitoring end-users

Load Balancing: Essential to the reliability and performance of a container-based application, load balancing optimizes workloads by fairly distributing them, thereby optimizing availability and avoiding undue system strain. For tasks that need parallel processing, load balancing breaks them into a fixed number of processes to be executed in parallel. The ephemeral nature of containers (they do not stay with the same host) complicates load balancing and makes service discovery necessary.

Machine Learning: With its potential for accurate detection, prediction and prevention, machine learning will be important to many areas, including security (detecting anomalies in user behavior, system communications, data transfers, or logins), infrastructure health monitoring (tracking server response times or network congestion), and business intelligence (measuring KPIs or business-critical IT services).

To solve for the operational complexity of each (and all) of the above, an enterprise must focus on its infrastructure architecture in tandem with modernizing its application architecture.

To Build or Not to Build?

> ***The choices for architecting a container-specific operational platform haven't changed from the choices for traditional infrastructures: Build or Buy.***

Either way, the end goal is an operational platform that's self-driving, self-healing, self-monitoring and able to handle storage, network and server provisioning, life-cycle management, capacity planning, performance monitoring, infrastructure scaling & upgrades, and security. (Oh, and, don't forget, a platform must maintain flexibility to future-proof as technology changes and, in kind, business strategy evolves.)

Architecting such an infrastructure requires a new way of thinking, one that goes beyond the typical enterprise cookbook solution. The traditional piecemeal approach to infrastructures must be replaced by a holistic approach that duly considers:

- **Weak Link:** Just as a chain is only as strong as its weakest link, an infrastructure will only be as strong as its weakest component. Thus, every service an infrastructure provides (orchestration, SDLC via CI/CD, and mission-critical services) must be optimized for containers.
- **Synergy:** The sum of an infrastructure's services must be greater than the sum of its parts. You cannot underestimate the importance of this: Every component of a containers infrastructure must work synergistically to produce a desired result.

With the above in mind, here's a closer look at each option.

Build

With today's open source software and widely available computing power, developers often have a do-it-yourself (DIY) mentality. But DIY open source must be carefully weighed: Is the enterprise able to sustain and maintain a DIY platform and withstand the impact of top engineers no longer focusing on the mainstream business? Before answering yes, consider:

1. The components required are often underestimated: Developers may be biting off more than they can chew. When taking on such a project, they often overlook features that help run software in production (such as consistently deploying and automating infrastructure, monitoring, and capacity planning).

2. Components are often not meant to work together: DIY platforms can require dozens of products. Even with solid open source tech with good APIs and useful documentation, connecting them isn't fast or easy. Often significant custom code must be written.

3. Building the platform is starting line, not the finish line: Running and maintaining it (consider upgrades and life-cycle of everything in production) is enormously complex.

4. Flexibility becomes difficult: Architectural shifts between on-premises resources and the public cloud are slow or costly.

5. Do you want to start your transformation in two years, or tomorrow? The focus must be time-to-market, time-to-value and impact to the business.

If that list alone doesn't shift DIY inclinations, then don't forget that while the aforementioned computing power may indeed be widely available, it can be costly at scale. This will leave an enterprise vulnerable to an unintended financial consequence when the infrastructure that was pieced together does not operate in a coordinated, efficient way (the earlier defined synergy). Before long, the enterprise (and its bottom line especially) will wish computing power had not been so available due to its cost at scale.

Buy

Current buying options typify how traditional infrastructures are built: piecemeal using an enterprise cookbook solution. Generally, vendor services fall into one of the categories below:

1. Containers-as-a-Service, or CaaS: Container engines, orchestration and the underlying compute resources are delivered as a service from a cloud provider; CaaS is often considered a type of IaaS.

2. Platforms as-a-Service, or PaaS: In addition to the features of CaaS, container-based PaaS solutions have build and deployment automation, service catalogs, runtime deployment support, scaling, orchestration, and monitoring.

3. Additional Tools & Services: While PaaS and Container platforms are capable of running both new and existing applications, many will need additional services in order to meet critical business needs.

As such, it's not uncommon for enterprises to piece together an infrastructure by choosing a CaaS/PaaS/IaaS provider with a vendor plug-in to handle the software application life-cycle and then selecting even more vendor products for mission-critical services.

Though this approach may pass the weak-link test (as each service may seem to be optimized for container-specific requirements), it may not pass the synergy test. And if it doesn't? The pain-points listed earlier may very well become daily IO reality. **The need for an infrastructure that functions synergistically cannot be overstated.**

To demonstrate the concept, we'll use two examples of capabilities that all containers infrastructures must have: the ability to scale, and the ability to remain future-proofed.

Scalability

> ***When viewed through the lens of synergy, an infrastructure's scalability will be created primarily by – and will rely on – the relationship and interconnection among its parts (which, in this scenario, are compute, resource utilization, and parallel processing).***

Individually, none ensures scalability: Parallel processing is only effective when resource utilization is precise and efficient. The same can be said for compute. Just as much as an infrastructure needs compute, it also needs to allocate it smartly via effective resource utilization. Thus, through the lens of synergy, an infrastructure's scalability will depend on whether it can dynamically and harmoniously harness its compute, resource utilization, and parallel processing in such a way that favors (i.e., enables) their combined action versus their individual component actions. If an infrastructure can do this, then scalability has been attained.

Future-Proofing

The need to future-proof is another reason synergy matters. For this example, data analytics is the context to demonstrate another nuance of synergy: the cooperative action of two or more stimuli will result in a different (or greater) response than that of the individual stimuli. Though this may not sound like much, make no mistake: The ability to future-proof relies on an infrastructure's ability to respond accordingly (and thus differently) to varying stimuli. Indeed, such versatility and adaptability is the only way an infrastructure will be able to handle the future requirements of evolving technologies.

In this data-analytics example, IoT data and machine learning are the 'stimuli.' Together, they will elicit a different (and greater) response from an infrastructure than individually. This is due to the interplay that will occur when processing IoT data and machine-learning algorithms, which will evolve due to the data.

1. Because IoT data is only useful when actionable, an infrastructure must be ready to take on–in real time–any and all spikes in structured, unstructured and semi- structured data. (Parallel processing will be key to this.) As soon as the real-time streams are processed, they must be analyzed immediately. For this to happen, an analytics capability must not just handle IoT speed and volume, but also, it must be able to support evolving developments in data analytics (which brings us to machine learning).

2. An evolving and fledgling technology that has vast potential applicability to a variety of problems, machine learning is also a compute-hungry technology. As adoption increases, and more sophisticated algorithms emerge (which will happen in response to analyzing IoT data, among other things), machine learning will place formidable demands on compute infrastructure.

> ***It is key that an infrastructure be able to adapt to future requirements such as this. The importance of ensuring an adaptable compute infrastructure cannot be overstated. If an infrastructure has this adaptability, then, and only then, can it qualify as future- proofed.***

Just as easily as a container can be spun up, so can dozens more examples similar to the above–all demonstrating why the components of a containers infrastructure must work synergistically. Without this capability, an infrastructure may not successfully scale to current needs, let alone remain future-proofed. An infrastructure architected a la carte with services from varying vendors may fail the synergy litmus test.

The IONATE Solution

With IONATE, an enterprise will skip the awkward, often painful development stages currently being experienced with poorly implemented containers infrastructure.

We do this through the IONATE platform, a unified container-based management platform that enables Dev and Ops personnel to collaboratively manage an application life-cycle from end to end, thus bringing true agility to a business. Combining serverless with supercompute, the IONATE platform guarantees compute, resource utilization and parallel processing, realizing tremendous benefits in scalability, efficiency and speed and transforming an existing infrastructure into a future-proofed supercompute infrastructure.

No other vendor on the market today combines serverless with supercomputing. By doing so, the IONATE platform provides the synergy necessary to optimize infrastructure health and performance and support all necessary services. **Our platform-agnostic solution allows an enterprise to scale and future-proof by orchestrating and optimizing machinery and processes.** Whether on the cloud or in the data center, we will transform an existing infrastructure into a supercomputing infrastructure, enabling higher utilization at a much lower cost **(often up to 90% of TCO reduction).**

> ***Finally, an infrastructure that doesn't just support business strategy. It is business strategy.***

We also support Kubernetes on IONATE Platform, which is much more secure, effective and high-performing when run on our platfrom. Enterprise applications run as serverless with built-in monitoring, security (including WAF for micorservices), persistence for SQL and NoSQL stores, and out-of-the-box AI Readiness for Spark and TensorFlow workloads.

To learn more about the IONATE Platform, and how IONATE can increase efficiency while increasing the bottom line, please contact us. #ionate