New technologies and practices follow a common adoption process in large enterprises. It typically begins with elaborated presentations and discussions, followed by proof of concepts. This leads to small-scale adoption within teams building non-critical systems, and mostly with their development and test workloads. By the time this technology reaches production, a significant amount of time and opportunities would have passed. This leads to increased opportunity cost, and friction between the development team's productivity and the organizational process.

Containers, and the growing containerization of workloads, is also going through a similar process in enterprises. Containers address the varied issues that plague developer feedback loops and infrastructure availability in enterprises.

A survey from Stack Engine shows that containers, and especially Docker, is primarily being used to get the most from the hypervisor investment. Not far behind for the adoption reason is the need to be infrastructure agnostic and be able to create test environments that do not take up too many resources. The results from the survey, although very encouraging, show it's still a long way before containers become a commonplace.

Motivation of Enterprises to Go for Containers

For an enterprise that is looking to evaluate this new trend, it would be interesting to note the indicators that gives rise to the evaluation of Containers:

1. Different teams in the Enterprise have their own standards of development and releasing software. This gets more pronounced when Enterprises outsource software development to third party vendors, often with limited governance and insight into their operations.

2. Legacy systems are a common place for an Enterprise. These Legacy systems create constant operational and maintenance issues. Project teams struggle to allocate resources to operate on-demand test infrastructure, and frequently lose out testing in favor of releasing functionalities in time.

3. Different teams have to calibrate software releases that usually take a long time, and increase the time to market. The calibration is necessary for a legacy system that is being continuously maintained and developed by multiple teams, sharing the same source trunk.

4. Production, Development and Test Environments have parity issues, and that leads to constant "runs-on-my-machine" syndrome among teams. Adopting VMs and Infrastructure automation has aided the situation a bit, but a large footprint that includes multiple VMs can be difficult to run on a development infrastructure.

5. Elastic Infrastructure based on VMs have become commonplace in Enterprise. However, VM sprawl is the common side-effect, thereby leading to inefficient

utilization of infrastructure. That leads to establishment of governance rules like Approvals and Workflows to limit the sprawl. But this reverses the benefits of Elasticity and Self service for Developers. Moreover unused VMs could lead to underutilized infrastructure, and increase the operational cost for enterprise.

6. Cloud infrastructure and new application architectures like Microservices are prone to slow adoption as it usually needs different view of infrastructure than traditionally assumed. Therefore, the teams in Enterprise lose out to create differentiators that could leverage new emerging trends.

Most of the indicators, as mentioned in the above list, are usually addressed by architectural changes, supported by an elastic on-demand infrastructure. Containers become the backbone to support these new architectural changes by providing abstraction around the workload and making it portable.

Iterations of Adoption

The first hints of the adoption of Containers takes place in the most obvious parts of the engineering cycle - The development and test environments. Ideally greenfield projects take the first plunge, but some curious ones also try to containerize their existing Projects to deal with Environment crunch. Basically, these existing projects have to deal with lack of infrastructure as one of the issues that plague their speed of execution. Containers becomes a good fix to resolve some aspects of this issue. Also, ramping up new development teams with project environments that are already containerized is far easier than traditional approaches. This helps to get the teams started with the Container technology and give them a good exposure by getting their hands dirty.

When Virtual Machines started getting popular inside Enterprise, a major sell to the IT Department was the goal to consolidate infrastructure, as a way to reduce the operational footprint, and hence costs. Containers and the ecosystem surrounding them, thanks to likes of Docker, however is not pitching the same goal. Teams are looking at Containers to truly make their apps portable, and not just to consolidate the infrastructure.

There are adoption steps that are common for enterprises evaluating Containers :-

1. Solving infrastructure availability is usually a good driver for moving up to Containers. Usually development teams have to fight out to get a Test Environment provisioned, and that takes a long amount of time. If dedicated instance is not needed, then reusing IT Infrastructure for hosting test environments as linked Containers is a good practice, and could lead to gaining operational knowledge

2. Build and Deployment infrastructure requires to be modified to leverage the generation of Container Images and deploying it to the respective infrastructure. Some Enterprises deploy the end application as a set of Machine images, instead of pushing the deployable on a running machine. This reduces the time to set up new

infrastructure if it has been provisioned just-in-time for the deployable. The same practice can be extended to Container images. Build step can generate the new Container image using a pre-existing Base image of the Environment. The Deployment step can take this image and run it on any infrastructure running the required Daemon like Docker.

3. Adopting a “Container-first” approach for all new projects is another common idiom that can help smoothen the adoption. This means all new projects will leverage Containers for their Build and Release process. This puts constraint on the development teams to consider containers as first class design element in their application topology and thereby making container native applications.

4. An important ingredient for Operations team is to formalize and release standard Container Images that all projects can use. These customized base images can then be hosted on a Private Registry that is used by Development teams when building projects. Changes to the standardized base images could then mean new releases on the registry which can then transparently inducted in the development process. Enterprises with their firewall rules are accustomed to host private repositories, and hence this will not be a strange adoption step for them. A Container-Image based development also helps more closer interaction between Development team and Operations, thereby encouraging DevOps.

Key Issues faced

Enterprise face two key challenges when adopting Containers in their overall IT strategy : Security, and the Lack of Mature tools in the Container Ecosystem. While most of the Web scale companies like Google, Twitter have been using Containers in production for many years, it’s still early days for Open Source and Proprietary products. The issue with tools is largely a time and experience bound problem. With more adoption and demand, it will inevitably be met.

Security on the other hand has been a dominant focus among peers who have been front runners of adoption. This is more of worry for companies who are already practising multi-tenancy with their production workloads.

> Container isolation has been a dreaded word for sometime, and quietly so has been in the center of discussion for long time.

Last year, Canonical introduced LXD as a way to implement ideas around stronger container security and make it mainstream with its Ubuntu and Openstack integration. New Projects like Hyper.sh that uses minimalist Linux kernel or HyperKernel are also emerging as interesting alternative to pure-play Containers.

As Docker progresses in its roadmap, we see a flurry of releases targeting Docker to be a Platform, and not just a tool any more. However the interesting part is that these platform features come as pluggable, and allows for replacement with any custom solutions if needed. Projects like Powerstrip from ClusterHQ are interesting options for Enterprise to test out various types of integration hooks with Docker daemon. This opens up a range of possibilities, and especially necessary for early movers who have to build wrappers for creating new use cases around Docker.

One of the other concerns that keeps coming inside Enterprise is the manageability aspects of Docker containers, especially when dealing with moving around containers in a large data center. Existing solutions like Apache Mesos with their Marathon API provides a clean way to perform container scheduling and resource management. Kubernetes from Google is also standing out as a plausible alternative thanks to growing community around it.

Containers in Practice

Last year, the Chief architect of ING spoke at Dockercon Europe about how they are leveraging Continuous Delivery around Docker. It was a sign of how enterprises are looking at Containers inside their organization. BBC news also highlighted how they architected their CI Solution around Docker with caveats and compromises.

AWS by releasing Elastic Container Service is making inroads in Enterprise, and providing a solid platform for organizations looking to move their existing public cloud workloads to container based environment. The release of Enterprise DockerHub last year was also an indicator that Docker is serious about Enterprise adoption.

Architecture style like Microservices are becoming prevalent in organizations, and Containers are becoming the de-facto deployment units of such services. At Build conference this year, Microsoft indicated integration of Docker natively inside Windows Platform, and with that it opens up Developers on the Microsoft platform to embrace Containers. This is clear a coming of age for Docker and Containers, inspiring the Enterprise to take notice and plan their strategy around this growing wave.

Vivek Juneja, an engineer based in Seoul, is focused on cloud infrastructure and microservices. He started working with cloud platforms in 2008, and was an early adopter of AWS and Eucalyptus Cloud. He's also a technology evangelist and speaks at various technology conferences in India. He writes @ www.cloudgeek.in and www.vivekjuneja.in, and loves grooming technology communities. You can also reach him by email: vivekjuneja@gmail.com.