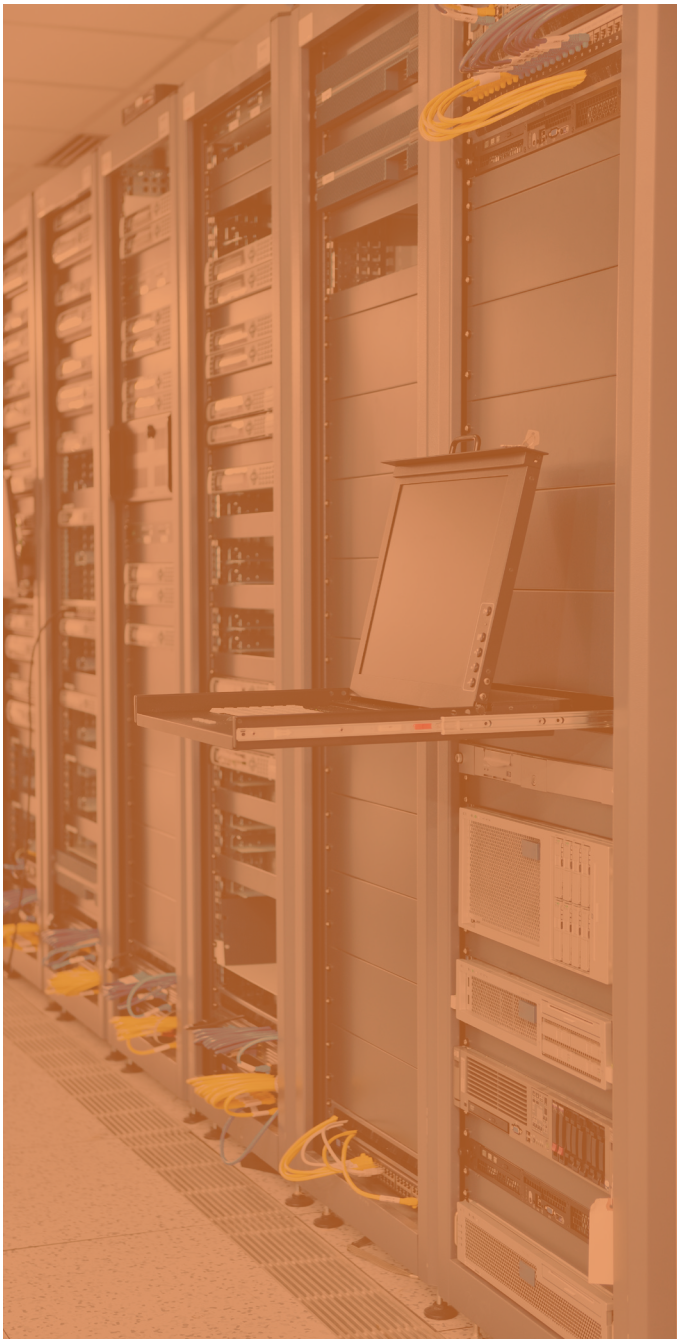
Mindtree

A PRACTICAL APPROACH TOWARDS ADOPTION OF CONTAINERS IN ENTERPRISE



Containers - A new way of software packaging (Code, libraries, runtime, system tools) in a lightweight, standard, secure manner. Containers are everywhere – Linux, Windows, Cloud, Data Centre Containers have captured the imagination of Enterprises. Latest trend reports, over 50% of companies have done some form of investment in container technologies. Container adoption have moved from pilot projects to large scale deployment. Enterprises are now using containers to run critical workloads in production. This paper touches upon few aspects on Container adoption at Enterprise scale, based on personal experience of delivering containerized environment for multiple clients

Why enterprise are adopting containerization technologies

- Introduce agility in business and technology
- Kickstart the Cloud Journey with Technology modernization
- Transform monolithic applications into microservices based architecture
- Get maximum value of Continuous Integration & Continuous Deployment
- Simplify Application Life Cycle management

What is container used for?

- Speedy deployments over multiple environment with consistency of code and configuration
- Promote concepts of Immutable Infrastructure
- Enhances the portability of application
- Infuse self-healing, and Auto scaling to introduce hands-off experience
  - Reduce expenses by optimizing resources and operations
  - Improves Uptime and reduces MTTR at the same time

Who uses containers:

DEVELOPERS

- Self Service with automated provisioning
- Flexible technical stack and application framework
- Build and run containers consistently across multiple environments (Dev, Test, Perf)
- Rapidly iterate, deploy to production faster

IT MANAGEMENT

- Run on any Infrastructure – Private, Public, Hybrid
- Better capacity utilization and control on infrastructure spend
- Enable standardization of process and patterns – Load Balancing, Secret management, deployment

BUSINESS

- Provides speed and Agility
- Time to market is reduced
- Lowering IT costs
- Reduction in Deployment Failure

Building Blocks of Container Ecosystem at Enterprise:

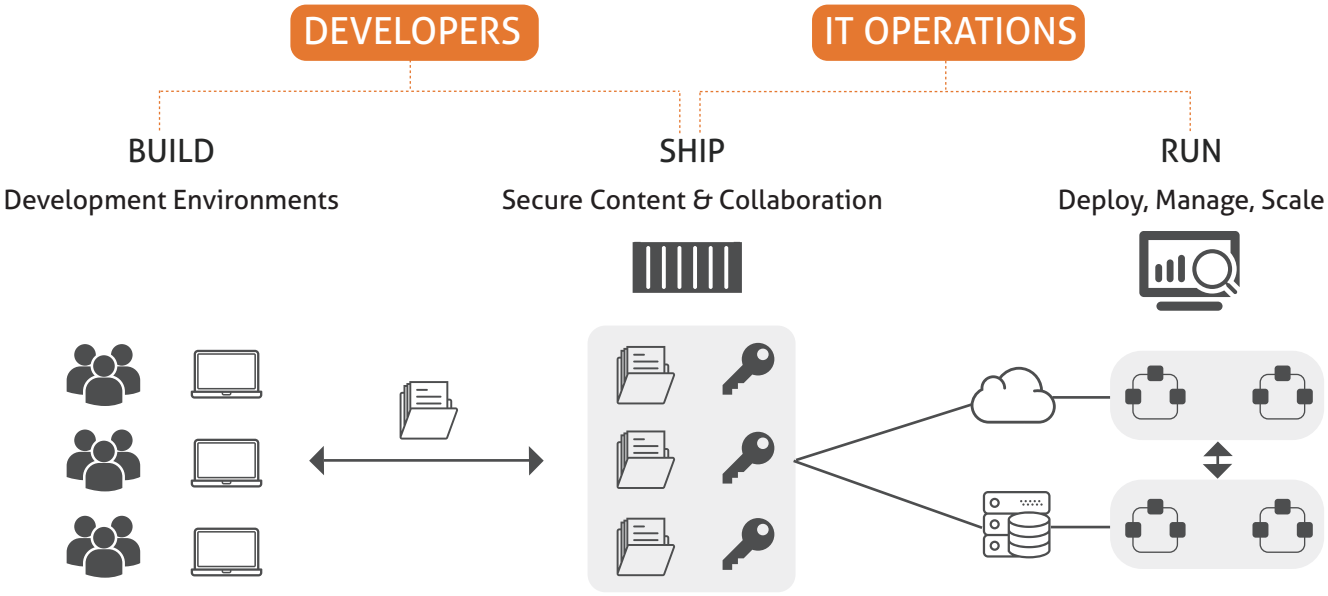
Container-As-a-Service – CaaS is a means or solution to achieve Containerization objectives:

CaaS is a service model that allows IT organizations to develop and deploy containerized applications. It is the container platform that handles the containerized application lifecycle

CaaS Features:

- Better manage application delivery
- Helps to build, ship, and run application anywhere
- Achieve consistency in Content and Infrastructure
- Agnostic to OS, Language, Infra stack
- Provide range of tools to manage / orchestrate complete life cycle of both Application and Infrastructures
- Live upgrade of running application without downtime
- Must support scaling of Application containers
- Support persistence of application state in a durable manner

Typical CaaS workflow:



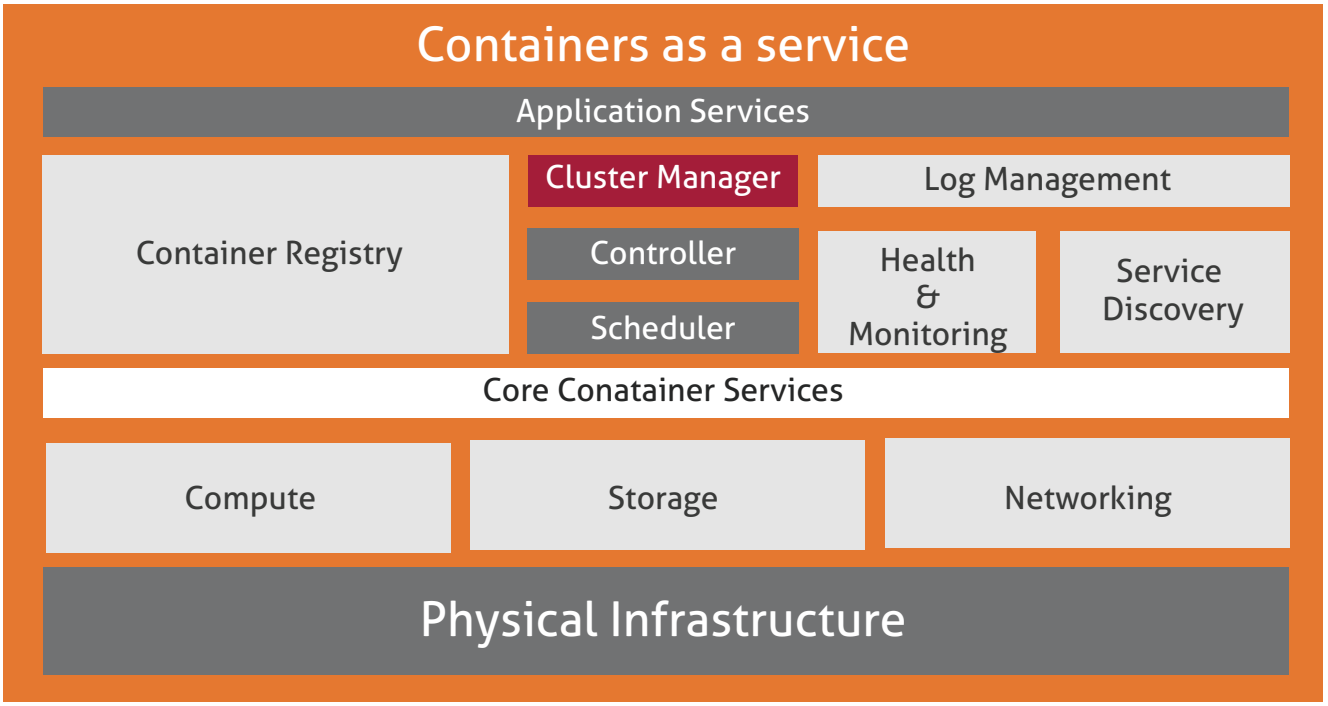
What does Container Technology offer?

It manages the lifecycle of containerized applications

- Workload Isolation, Service Discovery, Load Balancing, Configuration management
- Management of multiple Nodes in a cluster
- Scheduling of resources according to workload requirement
- Scaling of applications to a desired number of replicas
- Keeping health check record, self-healing
- User management, Secret management, authorization using RBAC model

Key Components of Container platform:

- Cluster Management
- Container Registry
- Container Services
- Scheduling



Implementation of Container Platform at Enterprise scale

Key Points to consider in implementation:

Understand Requirement:

- What kind of target environment (Linux, Windows etc.)
- Is it Pure cloud, or Hybrid stack to support
- Is it all going to be new development or need to connect with existing Infrastructure
- Workload Types, Availability requirement, Performance, Security, Volumes

Select and Understand Reference Architectures

– Kubernetes, AWS ECS/EKS, Mesosphere

Selection of Vendor/Technology for Container platform

- Kubernetes – Multi Cloud support, widely accepted in industry now
- Docker Swarm - Built on Native Docker technology
- AWS ECS - Elastic Container Service
- AWS EKS - Managed Kubernete service on AWS , New AWS service offering
- OpenShift - PaaS based on Kubernete
- Azure Container - Integration with Azure Services
- Mesosphere – Designed for large workloads

Enterprise Implementation of Container platform includes

- Service discovery
- Networking
- Security
- Centralized Logging
- Monitoring framework

Selection of Container platform:

Comparison between few popular container platforms in the market

Features	Kubernetes	AWS ECS	AWS EKS
Orchestration Technology	Open Source Kubernete	Amazon Service	Built On Open Source Kubernete
Deployment	On-Premise, Private & Public	Only on Amazon AWS EKS Platform	Only on Amazon AWS ECS platform
Container Type	Docker and ContainerD	Docker	Docker and ContainerD
Master Cluster Setup	Manual and Complex	Multiple Masters across AZ, managed by AWS	Comes with ECS Cluster
Workloads	Workload can run on any Kubernete Cluster	AWS ECS Cluster Only, Vendor Lock IN	Workload can run on any kubernete Cluster
Orchestration Capabilities	Rich features, Customizable	Limited	Rich features, Customizable
RBAC	Works , Supported by RBAC author	AWS IAM	AWS IAM
Application scalability constraints	Deployment definition supports both manual and automatic	Manual Scaling	Deployment definition supports both manual and automatic
Load Balancing within cluster	Exposed via services, can be used to load balance within	AWS ELB	Exposed via services, can be used to load balance within

Features	Kubernetes	AWS ECS	AWS EKS
Auto Scaling for applications	Supports by number of pods as well as resource metrics such as CPU, memory utilization and custom	Cloudwatch alarms, Lambda Functions	Supports by number of pods as well as resource metrics such as CPU, memory utilization and custom
Application Rolling upgrade/rollback	Yes	yes	yes
Block Storage	Flexible	AWS EBS	AWS EBS
Networking	Flat network model via overlay, Requires 2 CIDRs	AWS VPC	AWS VPC
Nodes per cluster	5000	1000	Mature, Flexible, ELK, FluentD, Cloudwatch
Logging	Mature, Flexible, ELK, FluentD	AWS Cloudwatch, CloudTrail	cAdvisor, sysdig, Prometheus/Grafanna
Monitoring	cAdvisor, sysdig, Prometheus/Grafanna	AWS Cloudwatch, CloudTrail, Partner tools Datadog, Sysdig	Multi AZ
Multi Data Centres/AZs	Kubernetes Federation(v1.9)	Multi AZ	YES
SSH access to Infrastructure	YES	YES	
Hybrid Cloud Support	YES	No	

Implementing Container at Enterprise level - Focus areas to consider

- Container Format – Standardizing container format
- Container Runtime – Supporting Container runtime
- Container Management - Ecosystem for deployment/app logic containerization
- Container Security consideration
- Orchestration and Control of Multi Cluster setups
- Aligning App Architecture and Deployment Architecture
- IaaS Integration and Abstraction
- Complex Multi Cloud support and Federation
- API and Gateway platform alignment
- Development standards, Tools, Language, Framework

Getting started Inputs while Implementing the Container platform in Enterprise

- Registry namespace practices – naming convention, organizational layout, taxonomy
- Docker image naming conventions
- Docker versioning conventions
- Authentication requirements
- Docker image registry hosting requirements
- When you can or can't proxy images from the public Docker hubs
- Docker image cleanup/garbage collection requirements (on a node, in the registry)
- When persistent volumes are appropriate and how they should be configured
- Requirements about the statelessness of container content – basically anything you write to a non-persistent volume in a container should be treated as perishable and throw-away-able
- Service discovery requirements – naming conventions when used with Kubernetes
- Logging requirements – how to log content in a container what to use to collect, index, search those logs

- Reviewing 12-Factor Industry characteristics - https://www.mirantis.com/blog/how-do-you-build-12-factor-apps-using-kubernetes/
- Review Monitoring framework/Tools

Common Challenges in a Multi Cluster setup of Containers in Enterprises

- Managing Upstream Kubernetes versions
- Standardizing Multi Cluster deployments
  - Versions, Networking, Ingress, Monitoring, Logging, Add-Ons
- Providing End to End Security
  - Image Scanning, Host & Cluster Scanning, Identity Management
- Centralized Application Management
  - Application Modelling and Definition
  - Environment – Application runtime management
  - Change Management
  - Application Monitoring, Logging
- Telemetry – Service Dependencies, Traffic Flow, Distributed tracing

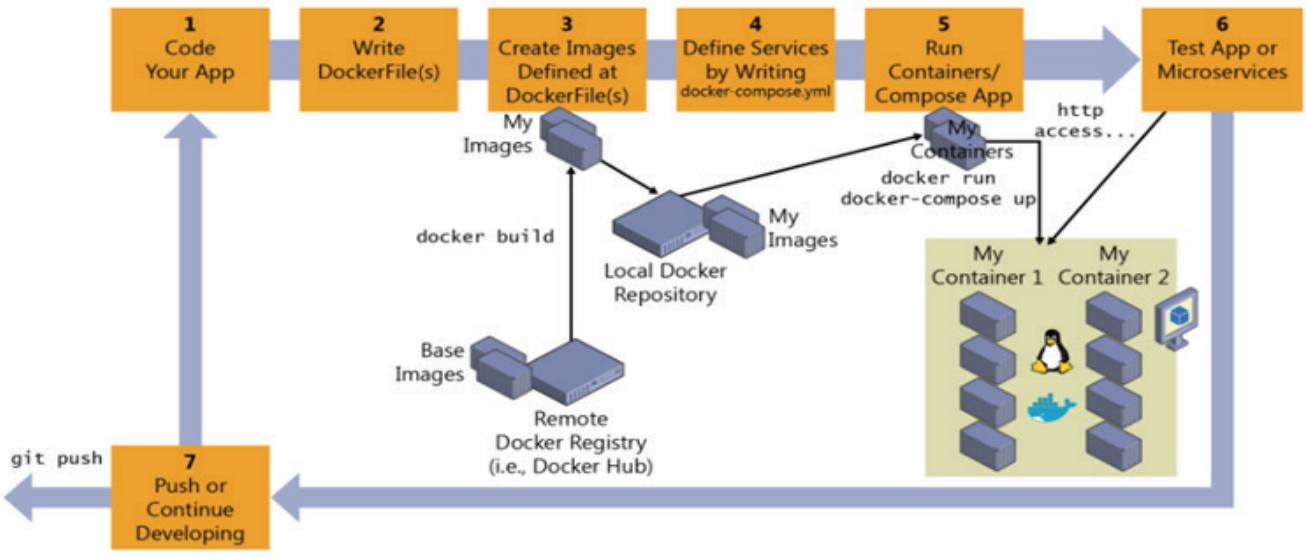
Few best practices implementing containers in Enterprises:

- Build Homogeneous Clusters: Offers similar capabilities and services for container workloads
  - Capabilities like External Load Balancer. Ingress Controller, Dynamically provisioned
  - Uniform K8S service configuration, RBAC policies
  - Support for add-ons like helm charts
- Decouple Cloud Infrastructure and K8S provisioning
- Bring up Consistent Infrastructure on multiple clouds., E.g. – Infra-as-a-code tools like Terraform work with most cloud providers
- Avoid Overlapping of IP Address across cloud in multi cluster setup
- Use the same Configuration management tool and scripts to setup all clusters
- Avoid Maintaining multiple resource definition for your clusters
- Ease of cluster operations and management – Similar add-ons, monitoring and Logging solution
- User Management: Integration with a common Identity provider, Common RBAC policies
- Uniform Namespace management for sharing clusters with different teams

Containerized Application Lifecycle Management:

How to manage container in production or at scale in Enterprise

Below diagram depicts a typical container application lifecycle



Container Security:

With increasing adoption of container at enterprise level and various application workloads running in container – It has become imperative to plan/design container security

- Only approved/certified content is running in production
- Industry and Government Standards and compliance are met
- Mitigation of External threat

Risks associated with container platform

- **Docker host and kernel security: Any attacker compromises your host system**

  Best Practice:
  - Make sure your host & Docker engine configuration is secure (restricted and authenticated access, encrypted communication, etc.
  - Subscribe to security for the OS and any software and install on the docker

- **Container breakouts: Docker container accessing sensitive information from the host bypassing isolation checks**

  Best Practice:
  - Remove Capabilities/access that are not required by your software
  - Do not run containers as uid 0
  - Create isolated user namespace limiting the maximum privileges of the containers over the host
  - Keep an eye on dangerous mountpoints from the host: the Docker socket (/var/run/docker.sock), /proc, /dev, etc

- **Container Image Authenticity: Pulling images without using any trust and authenticity mechanism**

  Best Practice:
  - Do not run unverified software and / or from sources you don't explicitly trust
  - Deploy a container-centric trust server
  - Enforce mandatory signature verification for any image that is going to be pulled or running on your systems

- **Compromised Secrets: Sensitive information compromised like user password hashes, server-side certificates, encryption keys**

  Best Practice:
  - Do not use environment variables for secrets
  - Do not embed any secrets in the container image
  - Deploy a Docker credentials management software if your deployments get complex enough

- Vulnerable container pay load – What if application running on container has vulnerabilities. Having generous logs and events from your service and hosted can greatly help in detecting the anomalies at container run time

Container Security - Indicative Reference Architecture

IMAGE SECURITY

- Image Verification
- Image Auditing
- CI/CD Integration
- Registry Security
- License scanning
- Static content scanning

RUNTIME SECURITY

- Cluster Integration
- Service based security
- Intelligent Policies
- Behavioral Threat detection
- Threat Protection

NETWORK SECURITY

- Network Policies
- Container Segmentation
- Network Enforcement
- DDOS, DNS attack protection
- Web Application Firewall

ENGINE SECURITY

Unprivileged deamon | Namespaces | Resource quotas

CONTAINER PLATFORM SECURITY

Rbac | Network encyrption | Volume encryption | Secret management

CONTAINERS + Microservices = Perfect Match for Enterprises

Microservices, Cloud Native applications are new generation application architecture. It decomposes the big monolithic application into small, discrete functions which can build and evolve independently

- Each Microservice is self-contained, Single business capability
- Each service is a separate codebase
- Each service persists their own data or external state
- Do not share data with other microservice
- Lightweight communication mechanism, HTTP resource API
- Independently Deployable
- Services don't need to share same technology stack, framework etc.
- Design for failure
- Decentralized Governance

Containers are natural fit for Microservice implementation. The container environment isolates between multiple services running on the same host, this avoids the risk of language, library, framework dependencies used by one microservice colliding with that of another. Containerized microservices are "Portable" across machines and providers, runs faster than VMs

Advantages of Microservice in containers:

1. Faster start time – Docker container starts in seconds, VM with complete OS takes minutes to load
2. Quicker Deployment – With Docker, we need to just download an image to run on any server
3. Easier management and scaling of containers
4. Better management of resources, more containers on a single server
5. Wider OS support – Docker for Debian, Mac, Windows etc.

Containers, Microservices, DevOps all combine very well to provide the Business and Technology Agility to Enterprises at scale. There are multiple paths to adopt containers, Every enterprise will have its own container adoption journey – Transforming Monolith to Microservices based architecture, Building Cloud Native apps, Adopting Hybrid Clouds.