

Introduction to Containers



**IBM Bluemix
Container Service**

Agenda

- Introduction
- Why containers?
- Docker containers
- Container orchestration
- Kubernetes
- Conclusion

Introduction



Everyone's container journey starts with one container....



At first the growth is easy to handle....





But soon it is overwhelming... chaos reigns



Regain control with Kubernetes

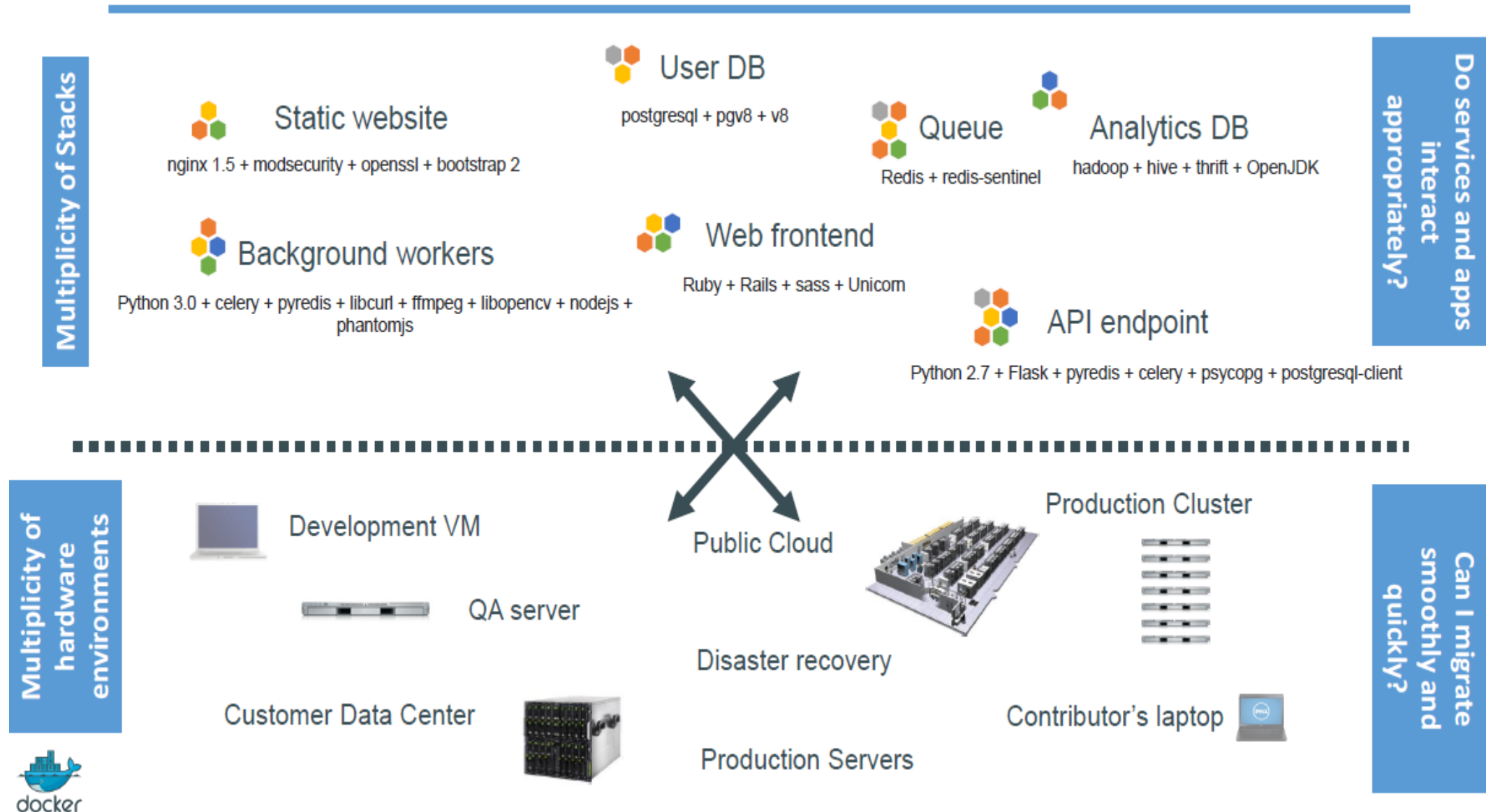




But let's not get ahead of ourselves...

Why Containers?

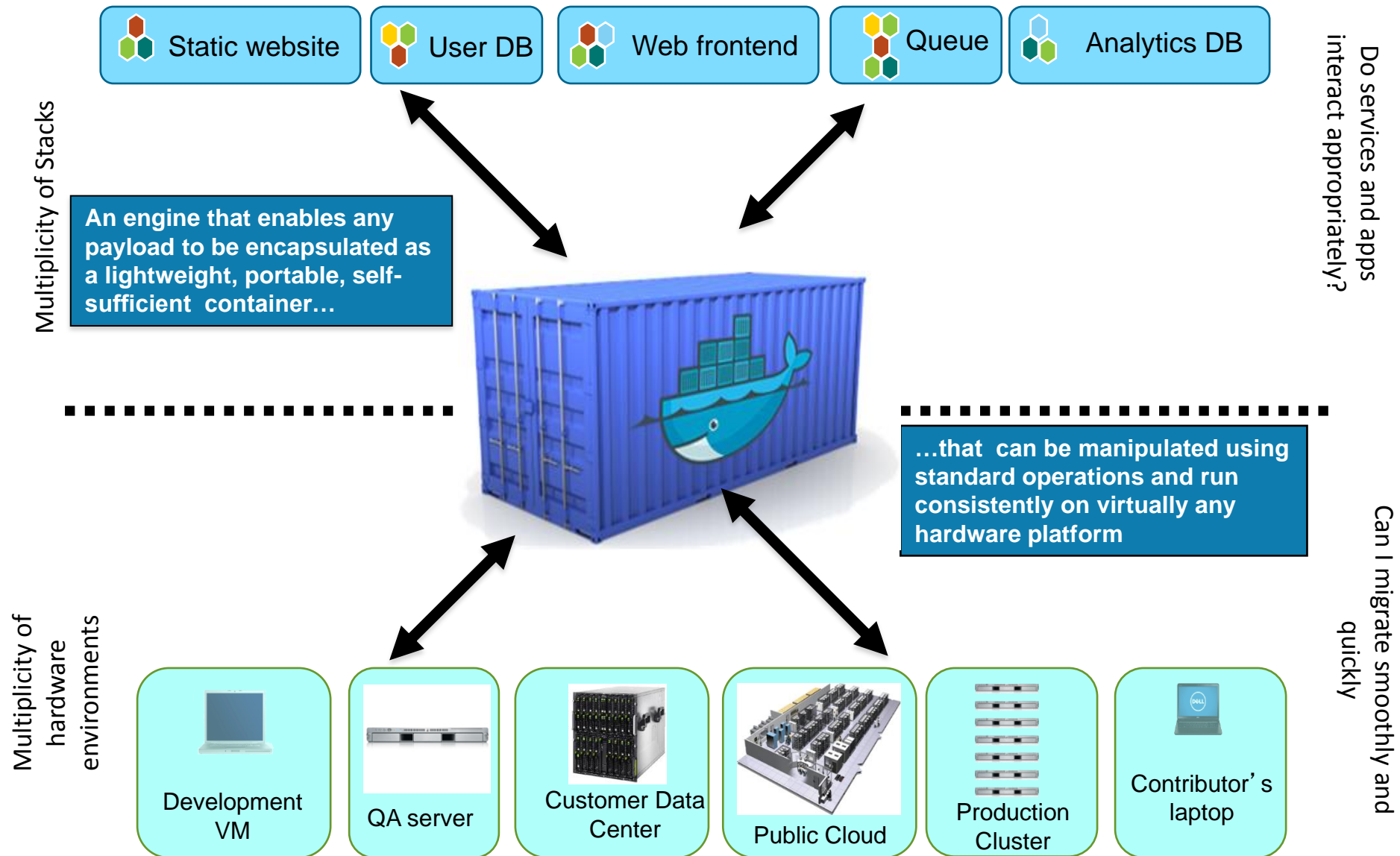
The Challenge



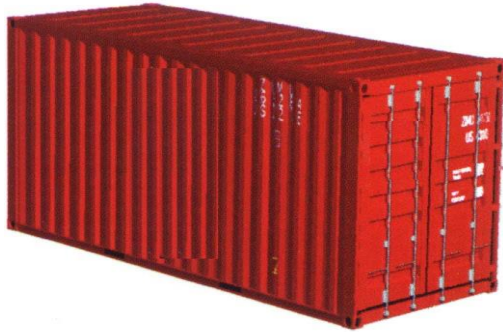
The Matrix From Hell

	Static website	?	?	?	?	?	?	?
	Web frontend	?	?	?	?	?	?	?
	Background workers	?	?	?	?	?	?	?
	User DB	?	?	?	?	?	?	?
	Analytics DB	?	?	?	?	?	?	?
	Queue	?	?	?	?	?	?	?
		Development VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor's laptop	Customer Servers

A Shipping Container for Code



Dev vs Ops

Dev**Ops**

- **Code**
- **Libraries**
- **Configuration**
- **Server runtime**
- **OS**

- **Logging**
- **Remote access**
- **Network configuration**
- **Monitoring**

Separation of concerns

A container separates and bridges the Dev and Ops in DevOps

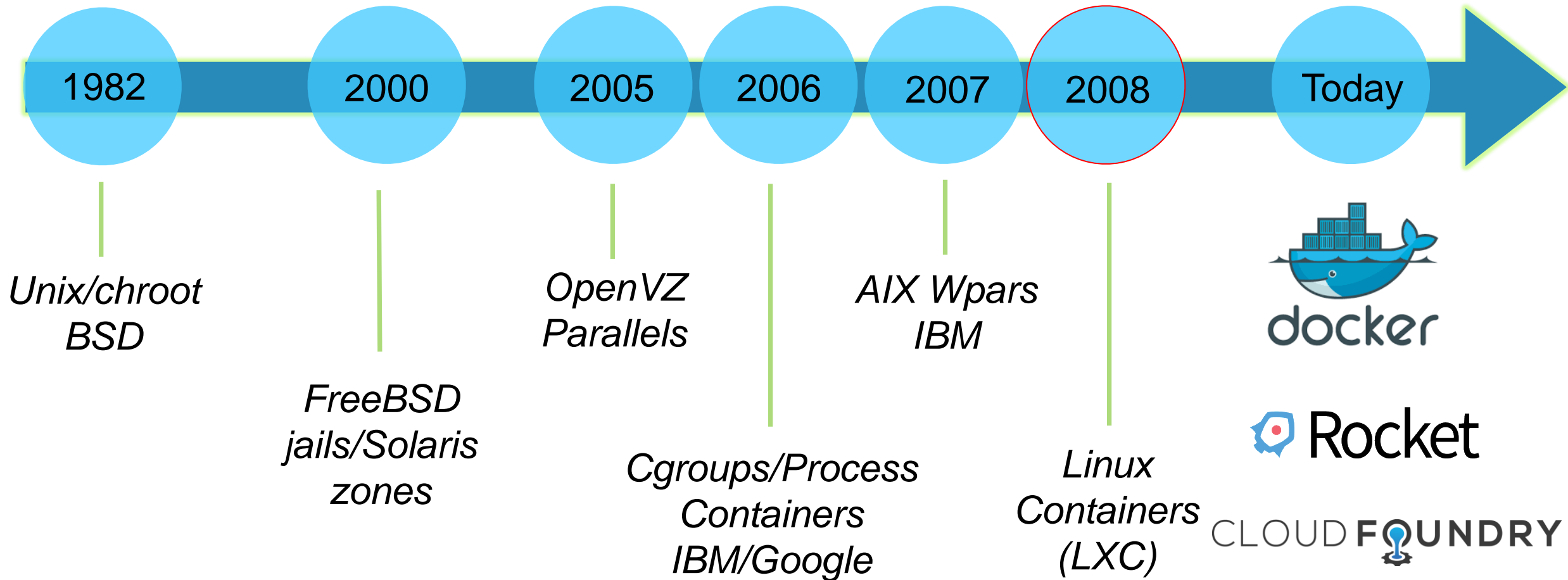
- Dev focuses on the application environment
- Ops focuses on the deployment environment

Everybody Loves Containers



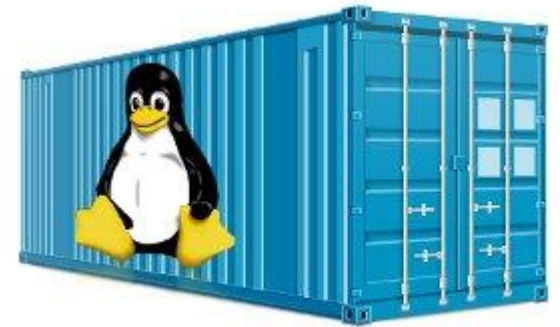
- A standard way to package an application and all its dependencies so that it can be moved between environments and run without changes
- Containers work by isolating the differences between applications inside the container so that everything outside the container can be standardized

Container History Lesson



Advantages of Containers

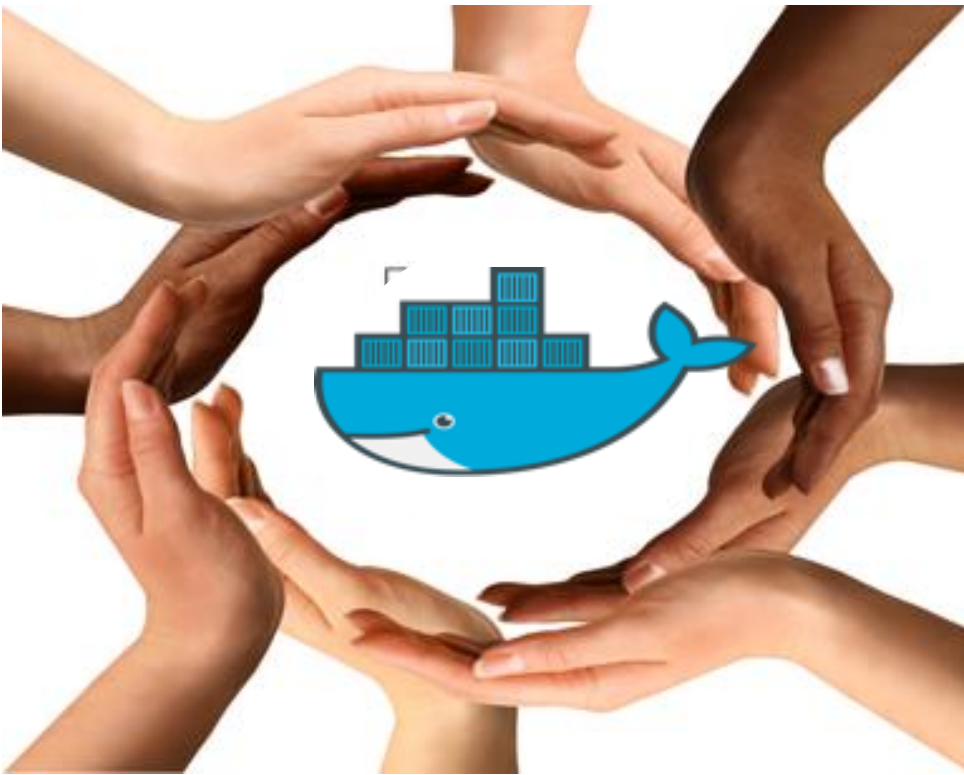
- Containers are portable
 - Any platform with a container engine can run containers
- Containers are easy to manage
 - Container images are easy to share, download, and delete
 - Especially with Docker registries
 - Container instances are easy to create and delete
 - Each container instance is easy and fast to start and stop
- Containers provide “just enough” isolation
 - More lightweight than virtual machines
 - Processes share the operating system kernel but are segregated
- Containers use hardware more efficiently
 - Greater density than virtual machines
 - Especially Docker containers, which can share layers
- Containers are immutable
 - Container images are versions
 - Containers cannot (should not) be patched



Docker Containers

Docker Adoption

Docker enables application development **efficiency**, making deployment more **efficient**, eliminating vendor 'lock-in' with true **portability**

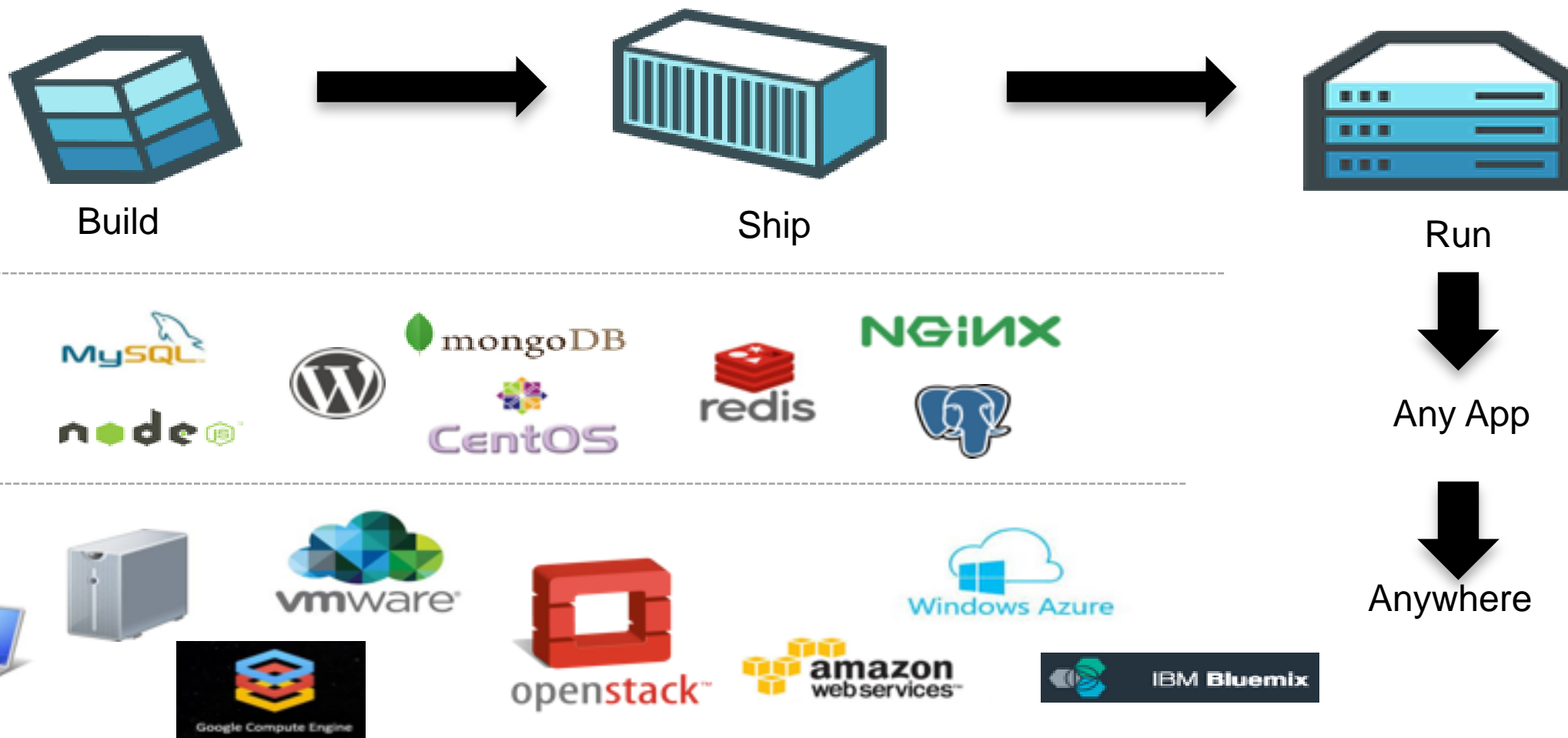


- Open software
 - Launched **March 2013**
 - 2.0+ billion downloads of Docker images
- Open contribution
 - 2000+ contributors
 - #2 most popular project
 - 185 community meet-up groups in 58 countries
- Open design
 - Contributors include IBM, Red Hat, Google, Microsoft, VMware, AWS, Rackspace, and others
- Open governance
 - 12 member governance advisory board selected by the community

Docker Mission



Docker is an **open platform** for building distributed applications for **developers** and **system administrators**

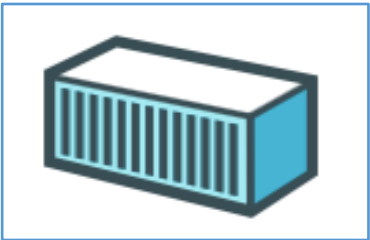


Docker Components



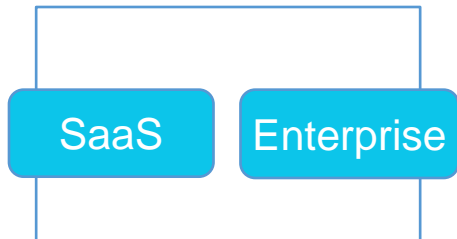
Image

- A read-only snapshot of a container stored in Docker Hub to be used as a template for building containers



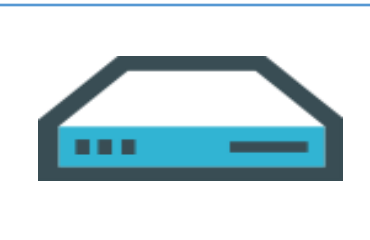
Container

- The standard unit in which the application service resides or transported



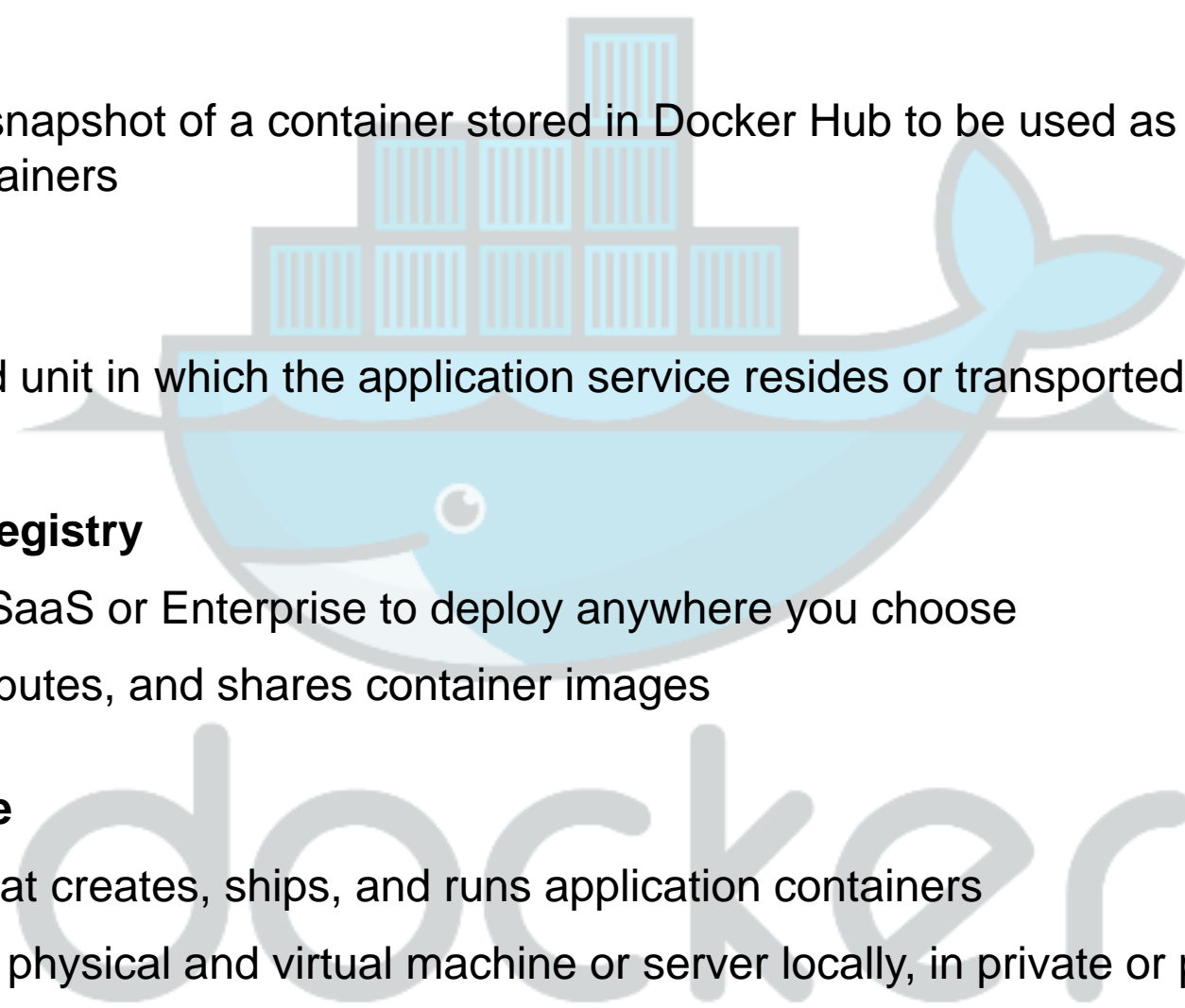
Docker Hub/Registry

- Available in SaaS or Enterprise to deploy anywhere you choose
- Stores, distributes, and shares container images

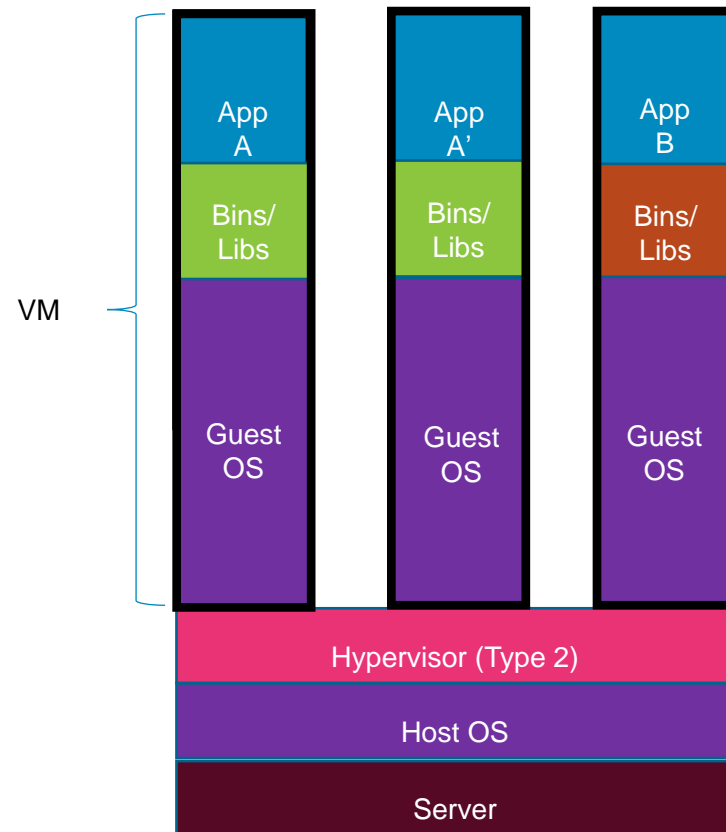


Docker Engine

- A program that creates, ships, and runs application containers
- Runs on any physical and virtual machine or server locally, in private or public cloud
- Client communicates with Engine to execute commands

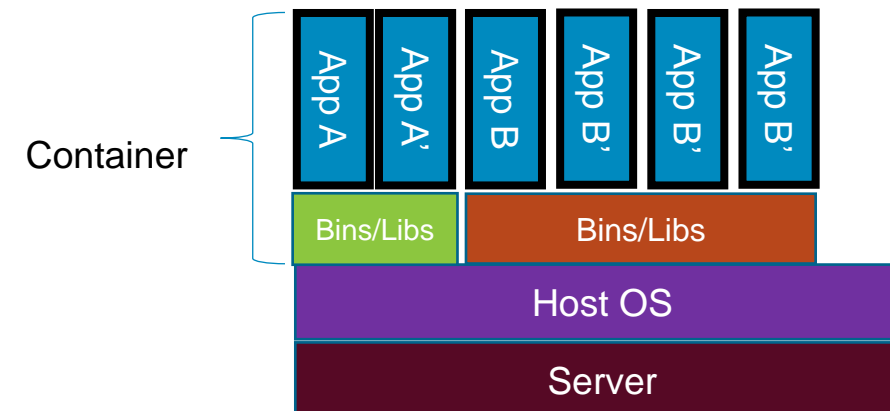


VMs vs Containers



Containers are isolated,
but share OS and, where
appropriate, bins/libraries

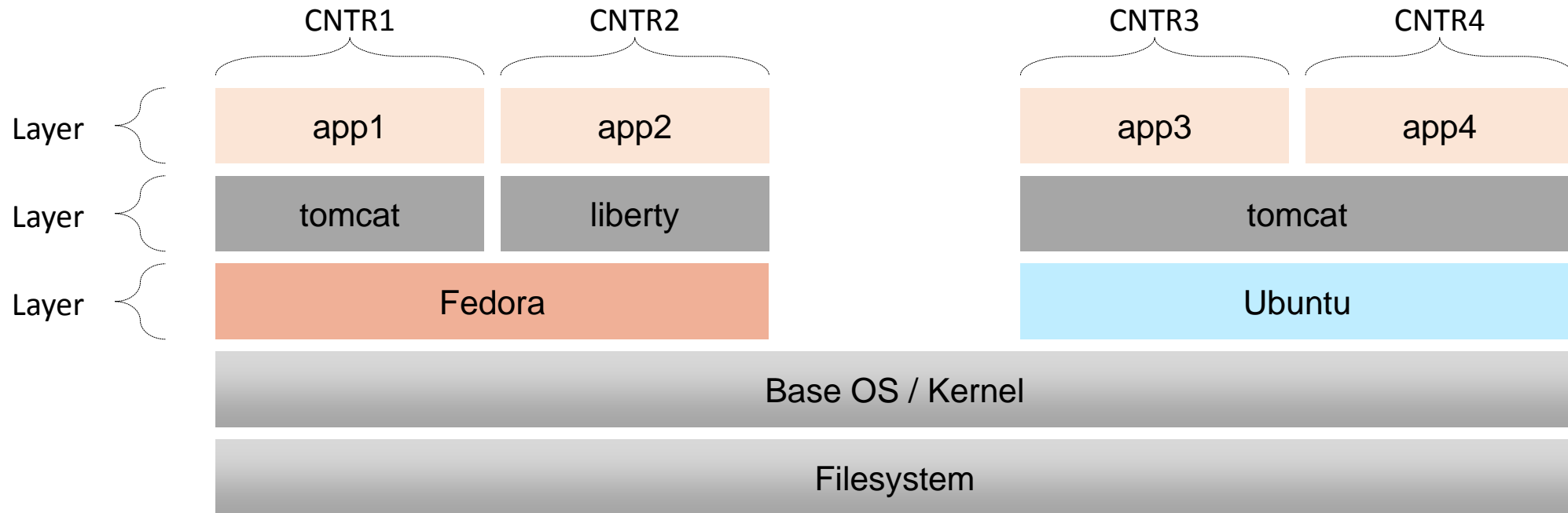
...faster, less overhead



Docker Containers



- Docker uses a copy-on-write (union) filesystem
- New files (& edits) are only visible to current/above layers



- Layers allow for reuse
 - More containers per host
 - Faster start-up/download time – base layers are "cached"
- Images
 - Tarball of layers (each layer is a tarball)

Why Customers are Interested in Docker Containers



- Containers are a critical foundation for hybrid apps
- Ship more software
 - Accelerate development and CI/CD pipelines by eliminating headaches of setting up environments and dealing with differences between environments
 - On average, Docker users ship software 7X more frequently
- Resource efficiency
 - Lightweight containers run on a single machine and share the same OS kernel while images are layered file systems sharing common files to make efficient use of RAM and disk and start instantly
- App portability
 - Isolated containers package the application, dependencies and configurations together
 - These containers can then seamlessly move across environments and infrastructures

Container Orchestration

More to Containers than just Docker

Serverless



PaaS



Container Orchestration



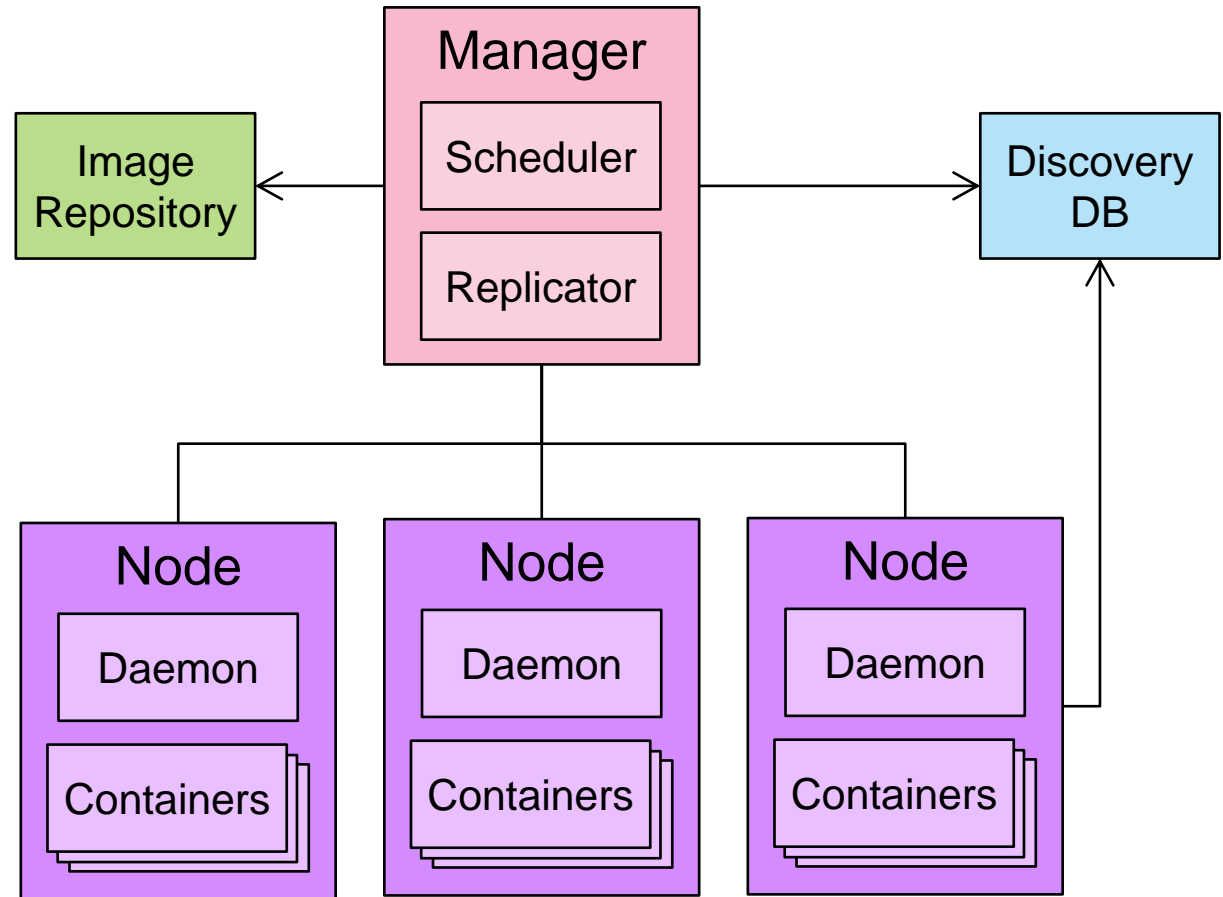
Container Engine



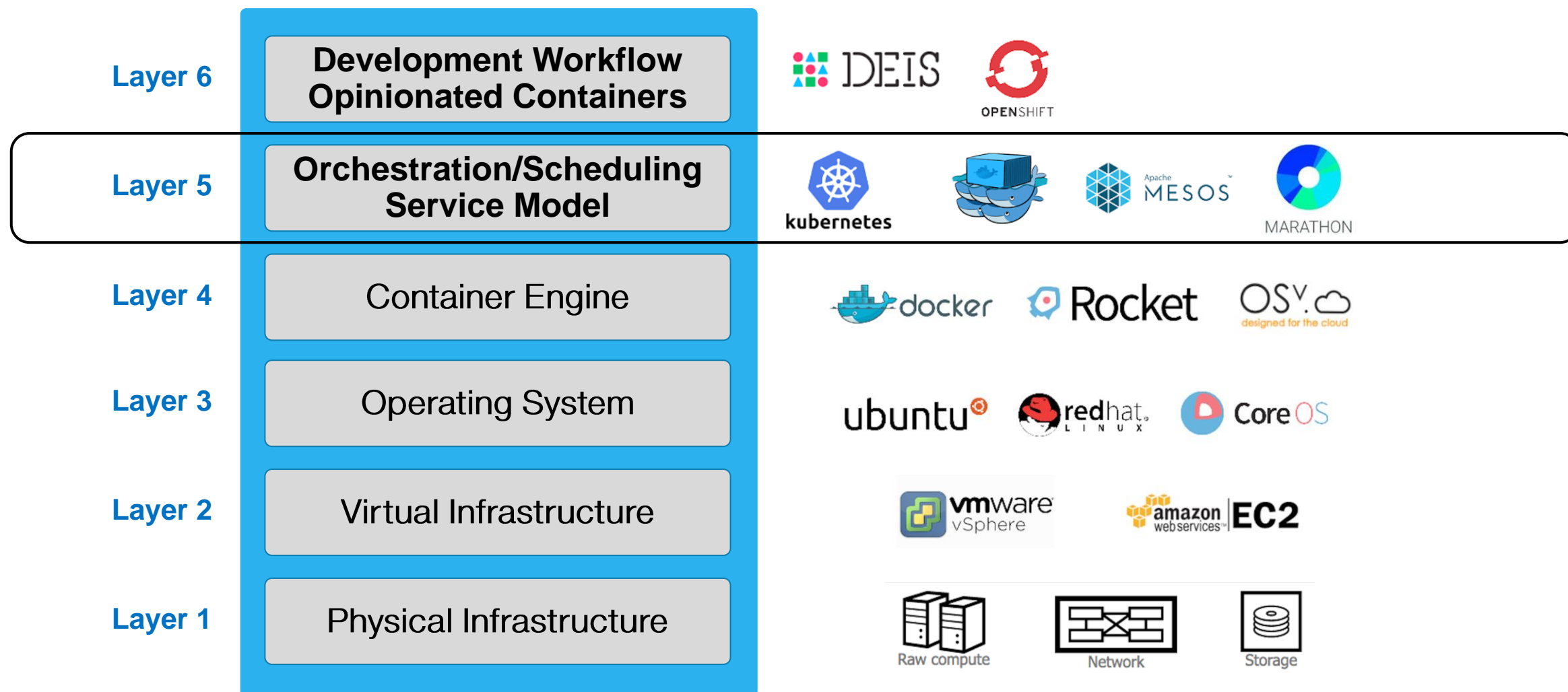
What is container orchestration?

- Container orchestration
 - Manages the deployment, placement, and lifecycle of workload containers
- Cluster management
 - Federates multiple hosts into one target
- Scheduling
 - Distributes containers across nodes
- Service discovery
 - Knows where the containers are located
 - Distributes client requests across the containers
- Replication
 - Ensures the right number of nodes and containers
- Health management
 - Replaces unhealthy containers and nodes

Container Orchestrator



Container Ecosystem Layers



Container Orchestration Responsibilities

- Container orchestration
 - Scheduling
 - Cluster management
 - Service discovery
- Related functionality
 - Provisioning
 - Monitoring
 - Configuration management

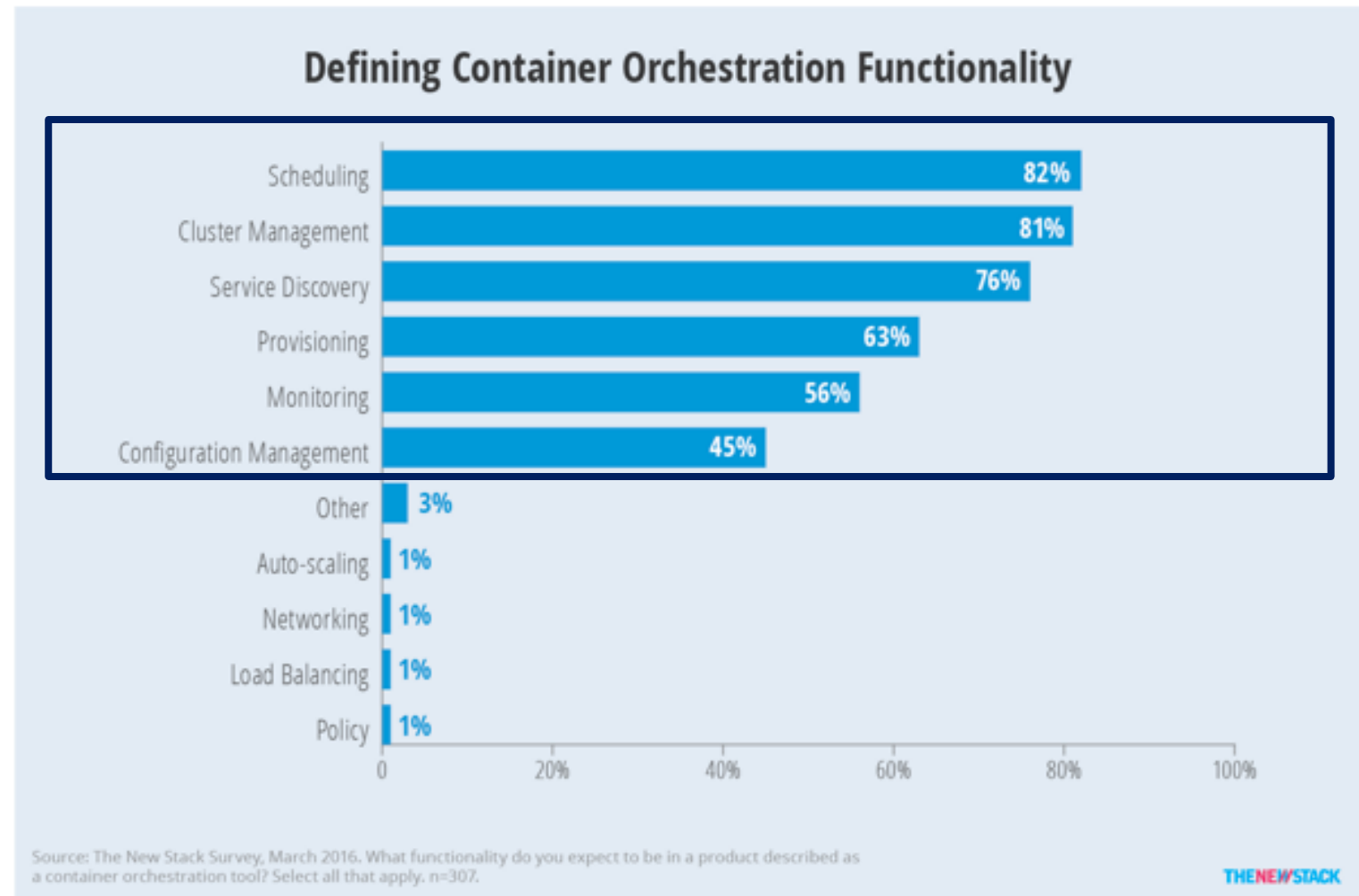


Figure 3: Only 45 percent of respondents consider configuration management to be part of a container orchestration product.

Container Orchestration in Bluemix

- In June 2015 when IBM Bluemix Container Service launched, open source orchestration projects were not available for production
 - Kubernetes – July 21, 2015
 - Docker Swarm – November 3, 2015
 - Apache Mesos – July 27, 2016
- IBM developed custom orchestration to ensure anti co-location of containers within a group and container placement to the least utilized hosts
- IBM Research runs rigorous performance and scalability testing (<https://github.com/Open-I-Beam/containers-os>) on various open source projects; then work with those communities to make improvements





Kubernetes

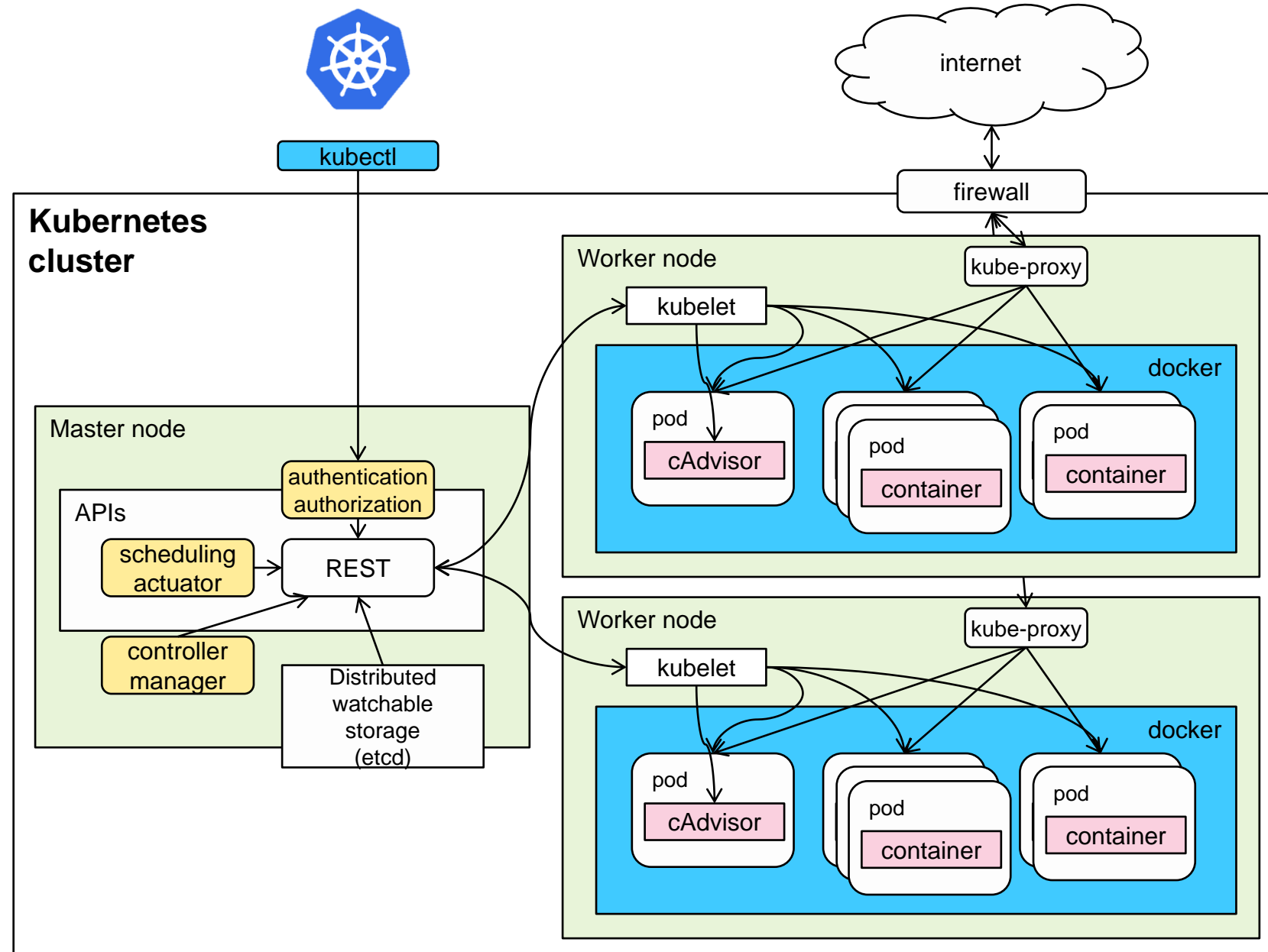
What is Kubernetes?



- Container orchestrator
 - Runs and manages containers
 - Unified API for deploying web applications, batch jobs, and databases
 - Maintains and tracks the global view of the cluster
 - Supports multiple cloud and bare-metal environments
- Manage applications, not machines
 - Rolling updates, canary deploys, and blue-green deployments
- Designed for extensibility
 - Rich ecosystem of plug-ins for scheduling, storage, networking
- Open source project managed by the Linux Foundation
 - Inspired and informed by Google's experiences and internal systems
 - 100% open source, written in Go

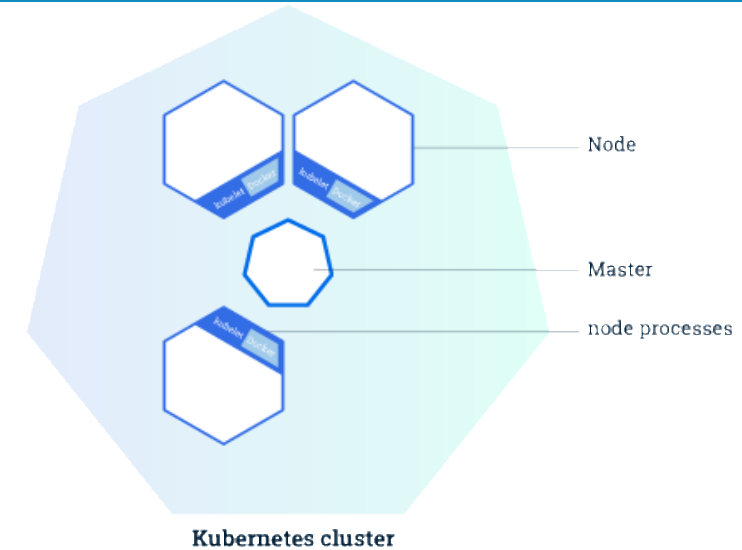
Kubernetes Architecture

- Master nodes
 - Computer with processes that manage the cluster
 - Multiple nodes for HA
- Worker nodes
 - Computers that host pods
 - Pods host application containers



Kubernetes Terminology: Topology

- Cluster
 - Collection of worker nodes managed by the same master node
 - Makes the worker nodes behave like one big computer
- Master node
 - Controls and manages the cluster
 - Scheduling and replication logic
 - Generally two or more master nodes for resiliency, but are not used for scaling out the cluster
- Worker node
 - Node where pods are run
 - Docker – Docker engine for running containers (including the kubelet)
 - kubelet
 - Kubernetes agent that accepts commands from the master
 - Manages pods in the node
 - cAdvisor – Container Advisor provides resources usage and performance statistics
 - kube-proxy – network proxy service responsible for routing activities for inbound or ingress traffic



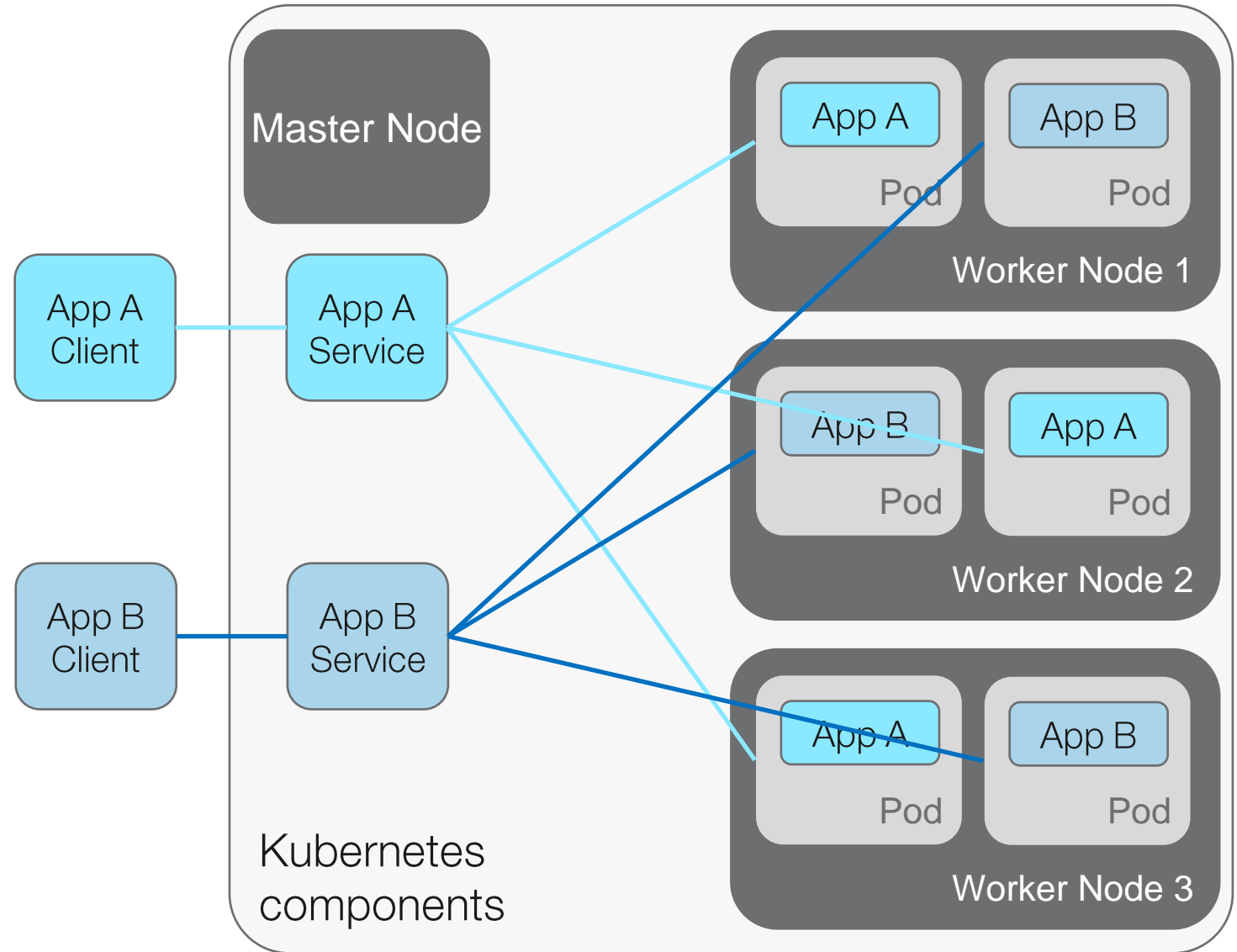
Kubernetes Terminology: Master Node Components



- Etcd
 - A highly-available key value store
 - All cluster data is stored here
- API Server
 - Exposes API for managing Kubernetes
 - Used by kubectl CLI
- Controller manager
 - Daemon that runs controllers, which are the background threads that handle routine tasks in the cluster
 - Node Controller – Responsible for noticing and responding when nodes go down
 - Replication Controller – Replaced by ReplicaSet
 - Endpoints Controller – Populates the Endpoints object (that is, joins services and pods)
 - Service Account & Token Controllers – Create default accounts and API access tokens for new namespaces
- Scheduler
 - Selects the worker node each pods runs in

Kubernetes Architecture: Workloads

- Container
 - Packaging of an app
- Pod
 - Unit of deployment
- Service
 - Fixed endpoint for 1+ pods



Kubernetes Terminology: Workloads

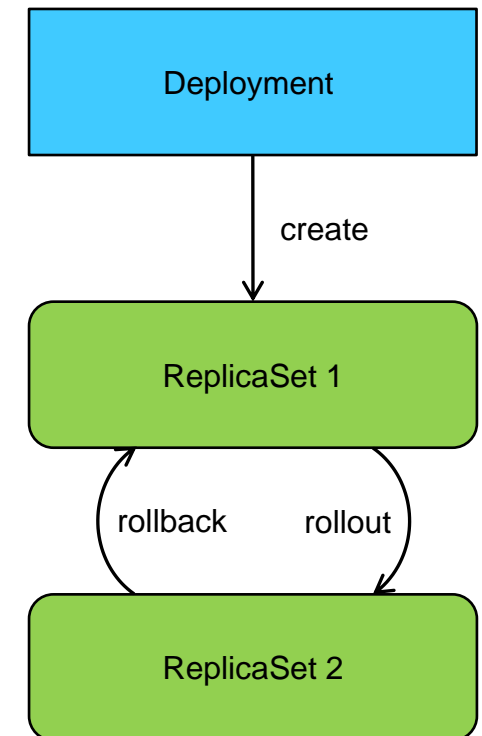
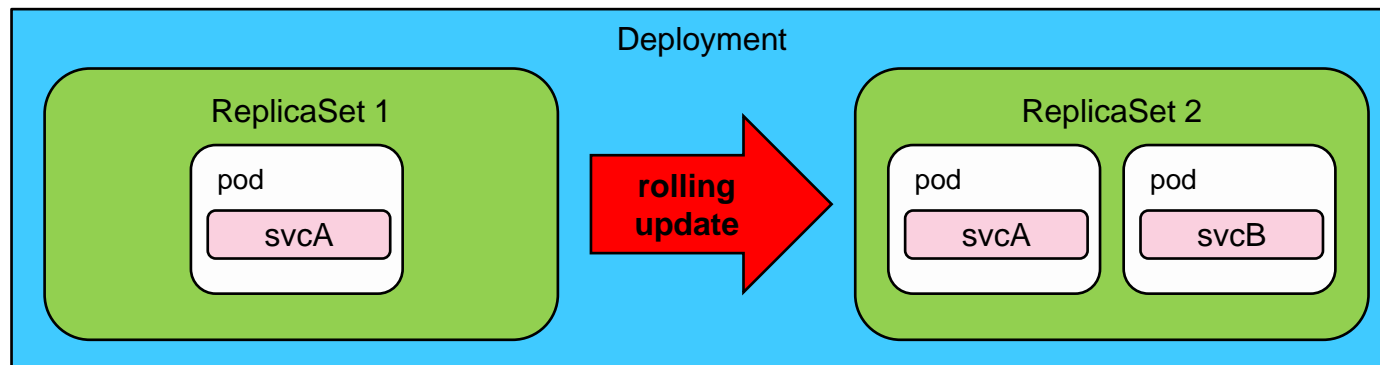


- Container
 - Unit of packaging
- Pod
 - Smallest deployment unit in Kubernetes
 - A collection of containers that run on a worker node
 - Each pod has its own IP
 - A pod shares a PID namespace, network, and hostname
- Service
 - Collection of pods exposed as an endpoint
 - Information stored in the Kubernetes cluster state and networking info propagated to all worker nodes
 - Types
 - ClusterIP – Exposes the service on a cluster-internal IP
 - NodePort – Exposes the service on each Node's IP at a static port
 - LoadBalancer – Exposes the service externally using a cloud provider's load balancer
 - ExternalName – Maps the service to an external name (such as foo.bar.example.com)

Kubernetes Terminology: Deployment



- Deployment
 - A set of pods to be deployed together, such as an application
 - Declarative: Revising a Deployment creates a ReplicaSet describing the desired state
 - Rollout: Deployment controller changes the actual state to the desired state at a controlled rate
 - Rollback: Each Deployment revision can be rolled back
 - Scale and autoscale: A Deployment can be scaled
- ReplicaSet
 - The next-generation ReplicationController
 - A set of pod templates that describe a set of pod replicas
 - Uses a template that describes specifically what each pod should contain
 - Ensures that a specified number of pod replicas are running at any given time



Kubernetes Autoscaling



- Horizontal Pod Autoscaling (HPA)
 - Automatically scales the number of pods in a replication controller, deployment, or replica set
 - Matches the observed average CPU utilization to the specified target
 - Fetches metrics in two different ways: direct Heapster access and REST client access
 - Kubernetes Heapster enables container cluster monitoring and performance analysis
 - Default config: query every 30 sec, maintain 10% tolerance, wait 3 min after scale-up, wait 5 min after scale-down

```
$ kubectl autoscale deployment <deployment-name> --cpu-percent=50  
--min=1 --max=10 deployment "<hpa-name>" autoscaled
```

- Creates a horizontal pod autoscaler
 - An HPA instance
 - Maintains between 1 and 10 replicas of the pods controlled by the deployment
 - Maintains an average CPU utilization across all pods of 50%

Kubernetes Terminology: Naming



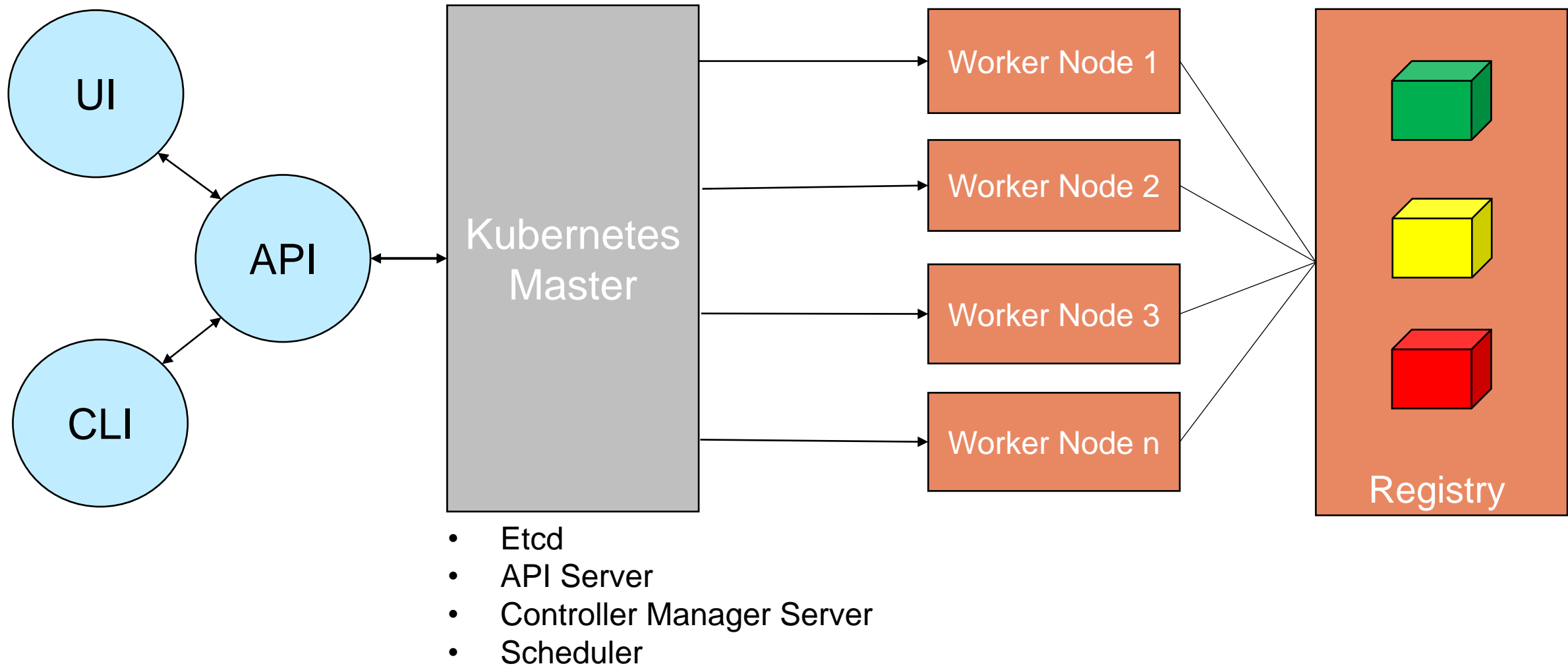
- Name
 - Each resource object by type has a unique name
- Namespace
 - Resource isolation: Each namespace is a virtual cluster within the physical cluster
 - Resource objects are scoped within namespaces
 - Low-level resources are not in namespaces: nodes, persistent volumes, and namespaces themselves
 - Names of resources need to be unique within a namespace, but not across namespaces
 - Resource quotas: Namespaces can divide cluster resources
 - Initial namespaces
 - `default` – The default namespace for objects with no other namespace
 - `kube-system` – The namespace for objects created by the Kubernetes system
- Resource Quota
 - Limits resource consumption per namespace
 - Limit can be number of resource objects by type (pods, services, etc.)
 - Limit can be total amount of compute resources (CPU, memory, etc.)
 - Overcommit is allowed; contention is handled on a first-come, first-served basis

Kubernetes Configuring Resources and Containers



- Label
 - Metadata assigned to Kubernetes resources (pods, services, etc.)
 - Key-value pairs for identification
 - Critical to Kubernetes as it relies on querying the cluster for resources that have certain labels
- Selector
 - An expression that matches labels to identify related resources
- ConfigMap
 - Configuration values to be used by containers in a pod
 - Stores configuration outside of the container image, making containers more reusable
- Secrets
 - Sensitive info that containers need to read or consume
 - Encrypted in special volumes mounted automatically

Kubernetes Architecture



Kubectl Commands

- Get the state of your cluster
`$ kubectl cluster-info`
- Get all the nodes of your cluster
`$ kubectl get nodes -o wide`
- Get info about the pods of your cluster
`$ kubectl get pods -o wide`
- Get info about the replication controllers of your cluster
`$ kubectl get rc -o wide`
- Get info about the services of your cluster
`$ kubectl get services`
- Get full config info about a Service
`$ kubectl get service
NAME_OF_SERVICE -o json`
- Get the IP of a Pod
`$ kubectl get pod NAME_OF_POD -
template={{.status.podIP}}`
- Delete a Pod
`$ kubectl delete pod NAME`
- Delete a Service
`$ kubectl delete service
NAME_OF_SERVICE`

Conclusion

Conclusion

- Why containers?
- Docker containers
- Container orchestration
- Kubernetes

Resources

- Docker tutorial
 - <https://docs.docker.com/get-started/>
- Kubernetes tutorial
 - <https://kubernetes.io/docs/tutorials/kubernetes-basics/>
- The Evolution of Linux Containers and Their Future
 - <https://dzone.com/articles/evolution-of-linux-containers-future>
- Introduction to container orchestration
 - <https://www.exoscale.ch/syslog/2016/07/26/container-orch/>
- TNS Research: The Present State of Container Orchestration
 - <https://thenewstack.io/tns-research-present-state-container-orchestration/>
- Large-scale cluster management at Google with Borg
 - <https://research.google.com/pubs/pub43438.html>