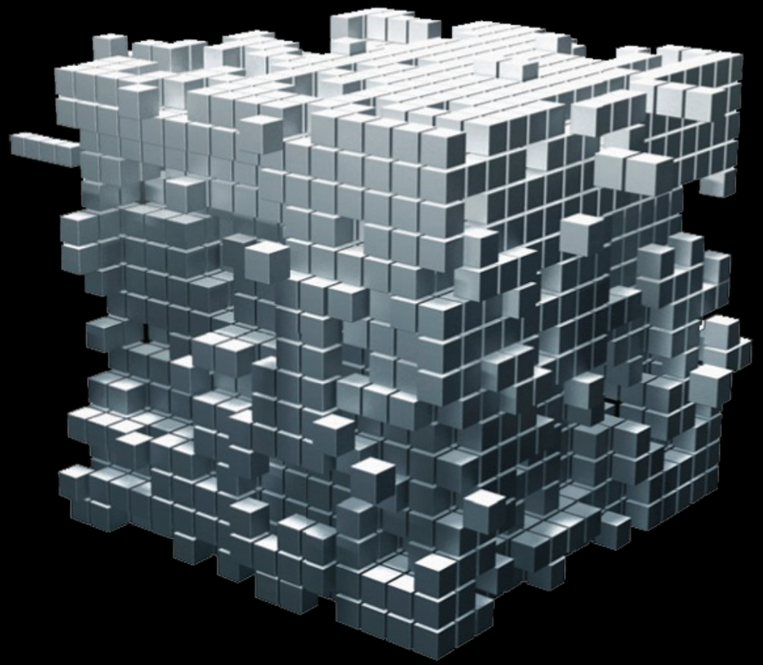
Characterizing and Contrasting Container Orchestrators

Lee Calcote

http://calcotestudios.com/ccco

Lee Calcote

clouds, containers, infrastructure, applications and their management



@lcalcote

blog.gingergeek.com

linkedin.com/in/leecalcote

lee@calcotestudios.com

Definition:

[k*uh* n-**tey**-ner]

[**awr**-k*uh*-streyt-or]

~~Fleet~~

~~Nomad~~

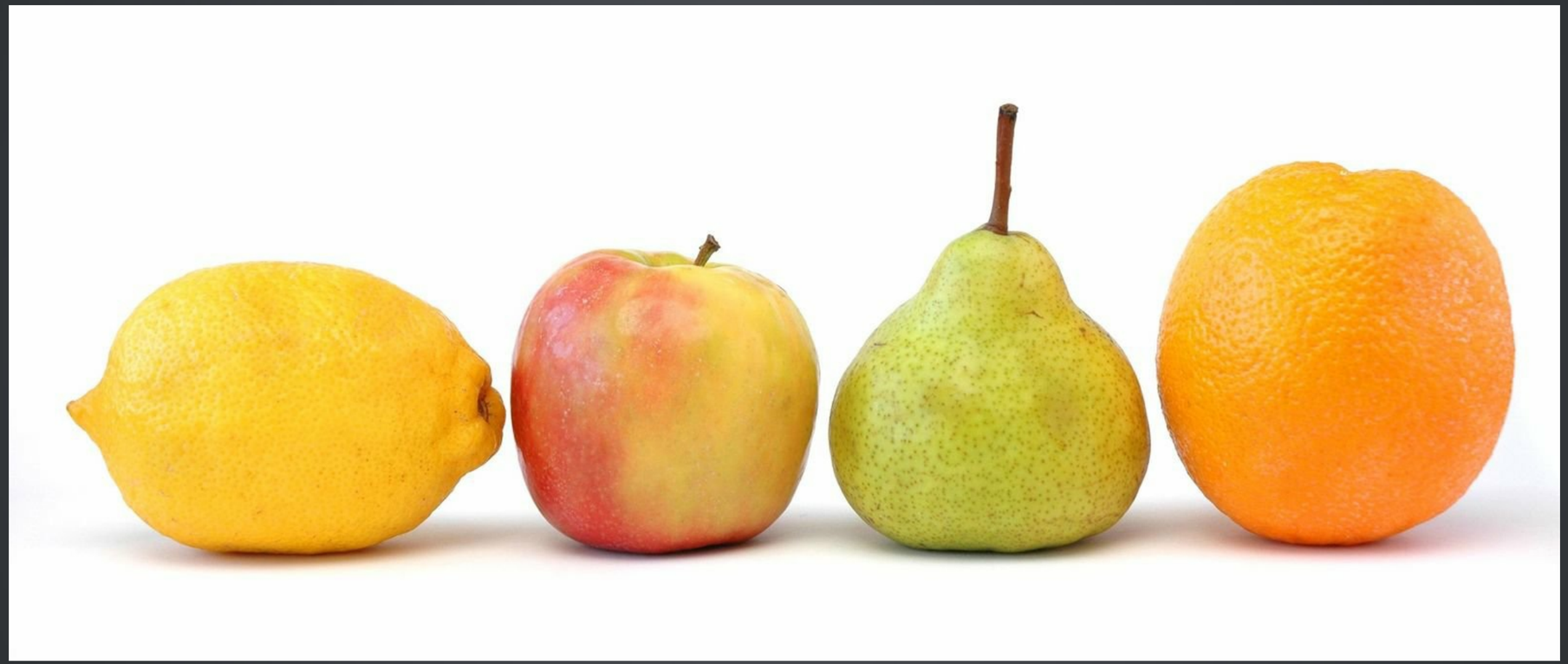
Swarm

Kubernetes

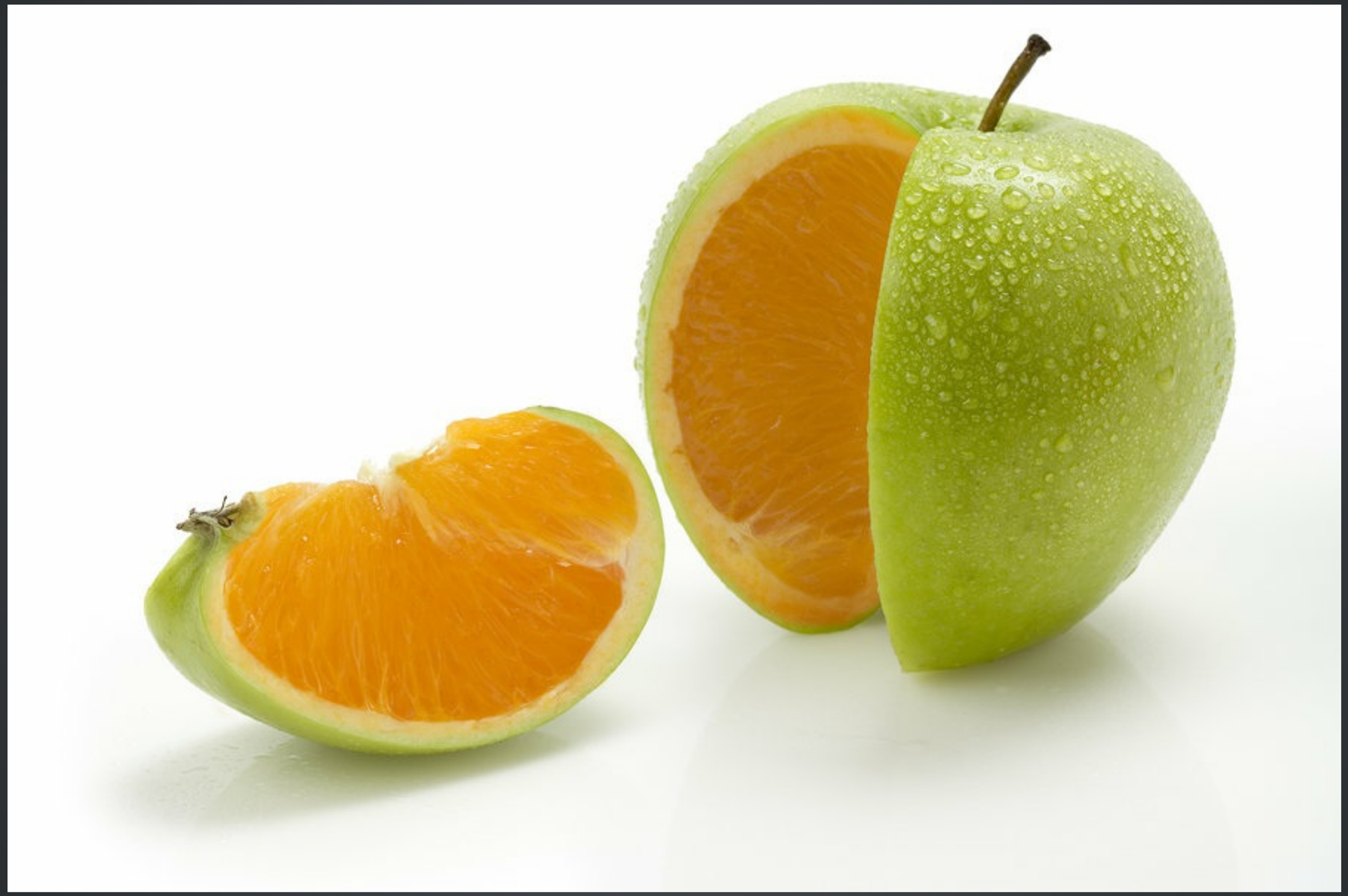
Mesos+Marathon

(Stay tuned for updates to presentation)

A strict apples-to-apples comparison is inappropriate and not the objective, hence characterizing and contrasting.



One size does not fit all.



Categorically Speaking

Categorically Speaking

- Genesis & Purpose

Categorically Speaking

- Genesis & Purpose
- Support & Momentum

Categorically Speaking

- Genesis & Purpose
- Support & Momentum
- Host & Service Discovery

Categorically Speaking

- Genesis & Purpose
- Support & Momentum
- Host & Service Discovery
- Scheduling

Categorically Speaking

- Genesis & Purpose
- Support & Momentum
- Host & Service Discovery
- Scheduling
- Modularity & Extensibility

Categorically Speaking

- Genesis & Purpose
- Support & Momentum
- Host & Service Discovery
- Scheduling
- Modularity & Extensibility
- Updates & Maintenance

Categorically Speaking

- Genesis & Purpose
- Support & Momentum
- Host & Service Discovery
- Scheduling
- Modularity & Extensibility
- Updates & Maintenance
- Health Monitoring

Categorically Speaking

- Genesis & Purpose
- Support & Momentum
- Host & Service Discovery
- Scheduling
- Modularity & Extensibility
- Updates & Maintenance
- Health Monitoring
- Networking & Load-Balancing

Categorically Speaking

- Genesis & Purpose
- Support & Momentum
- Host & Service Discovery
- Scheduling
- Modularity & Extensibility
- Updates & Maintenance
- Health Monitoring
- Networking & Load-Balancing
- High Availability & Scale

Hypervisor Manager Elements

- Compute
- Network
- Storage

Container Orchestrator Elements

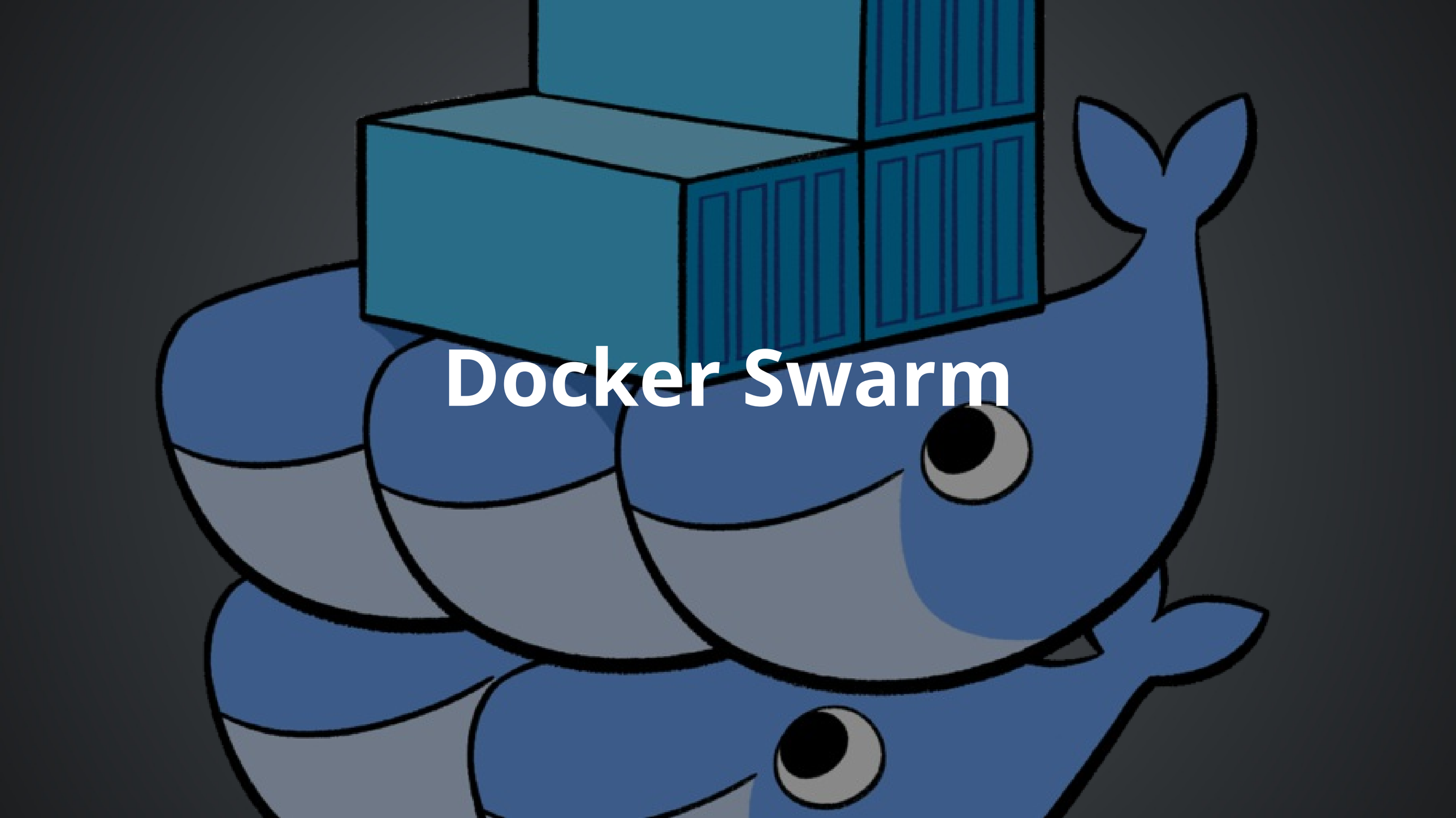
- Host (Node)
- Container
- Service
- Volume
- Applications

Core Capabilities

- Scheduling
- Cluster Management
  - Host Discovery
  - Host Health Monitoring
- Orchestrator Updates and Host Maintenance
- Service Discovery
- Networking and Load-Balancing

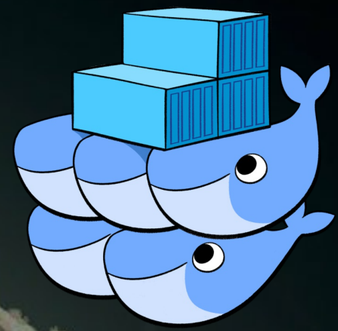
Additional Key Capabilities

- Application Health Monitoring
- Application Deployments
- Application Performance Monitoring

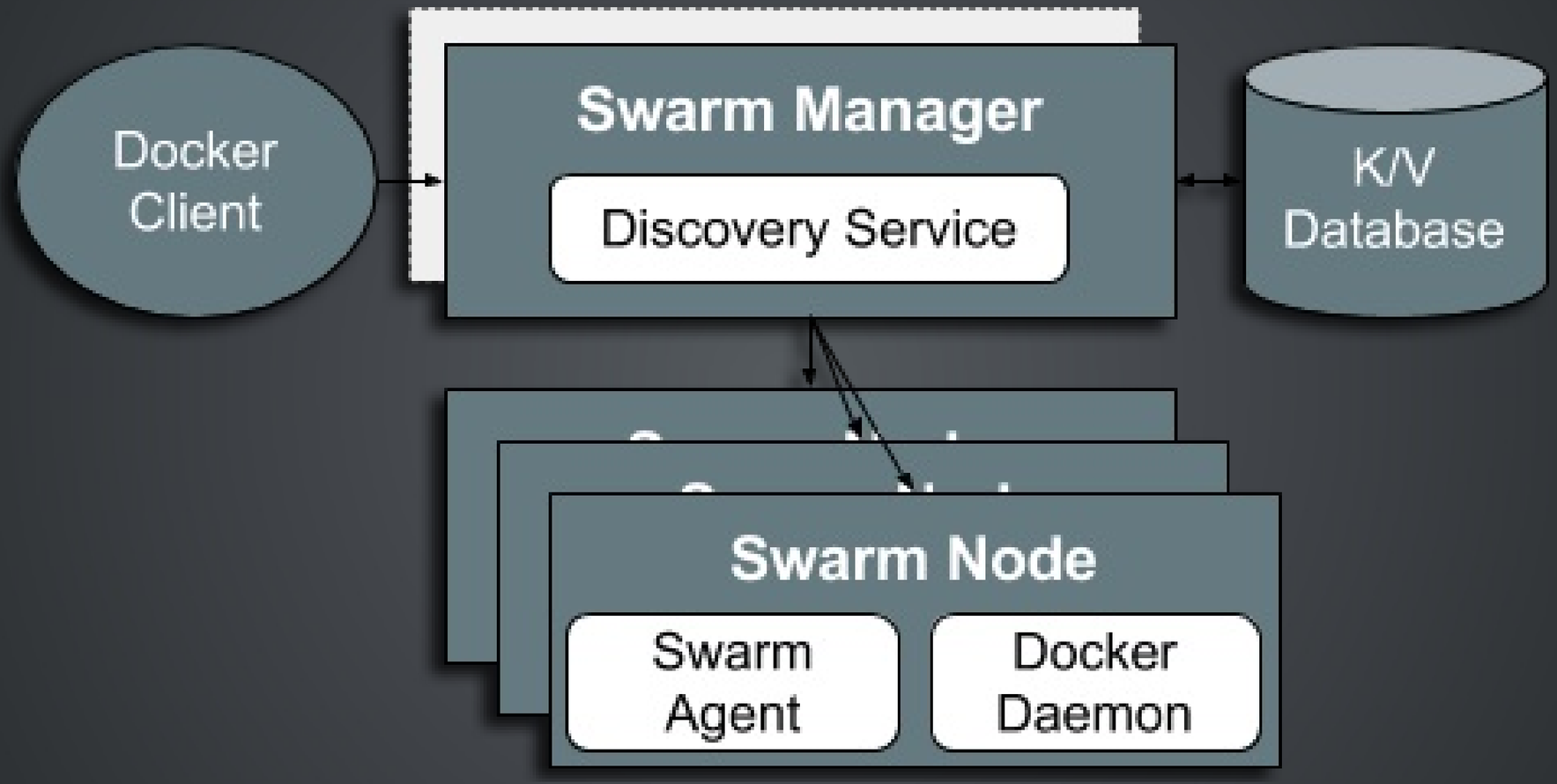


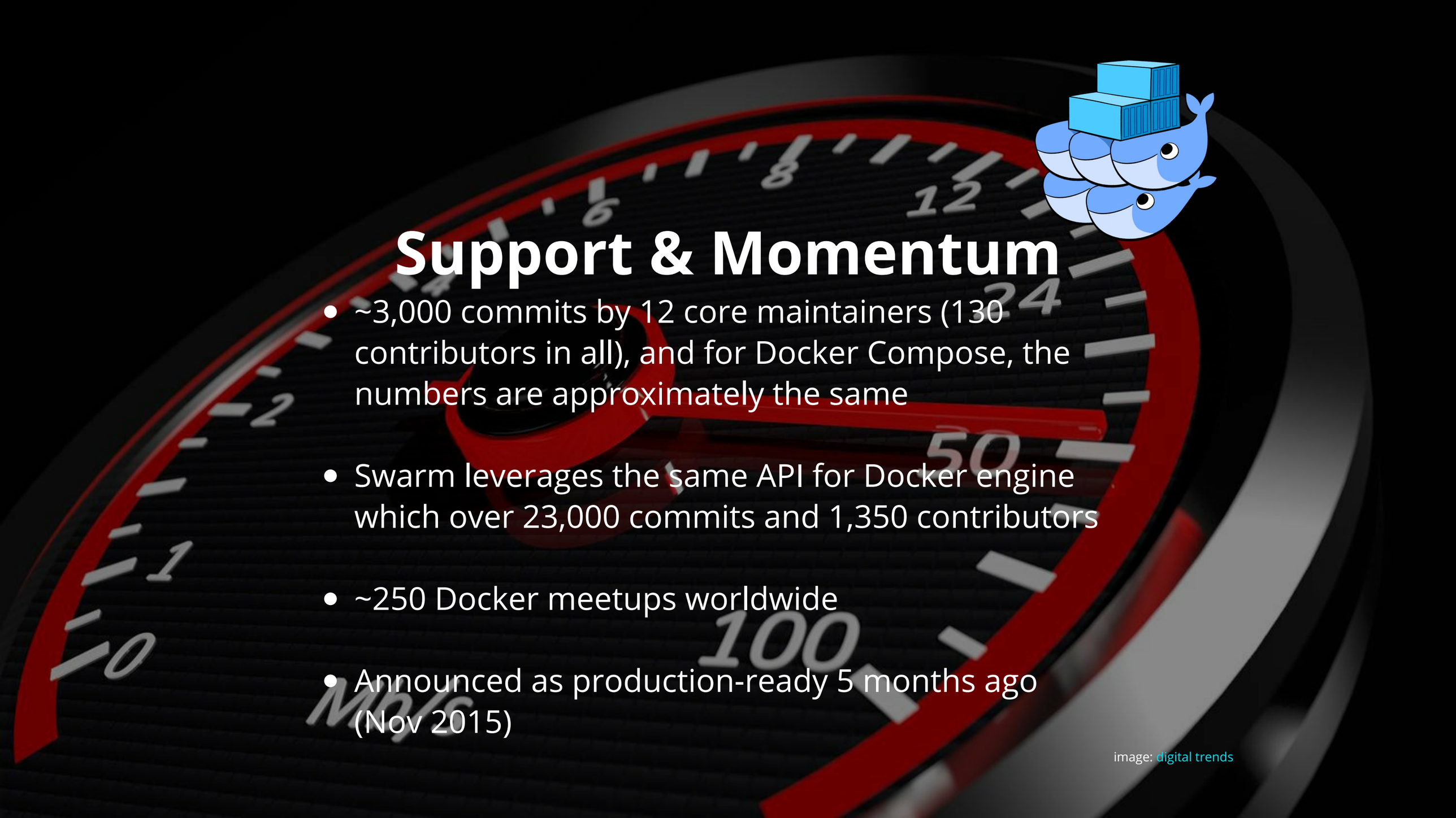
Docker Swarm

Genesis & Purpose



- Swarm is simple and easy to setup.
- An imperative system, Swarm is responsible for the clustering and scheduling aspects of orchestration.
- Swarm’s architecture is not complex as those of Kubernetes and Mesos
  - has Manager(s) and Agent(s)
- Written in Golang, Swarm is lightweight, modular, portable, and extensible

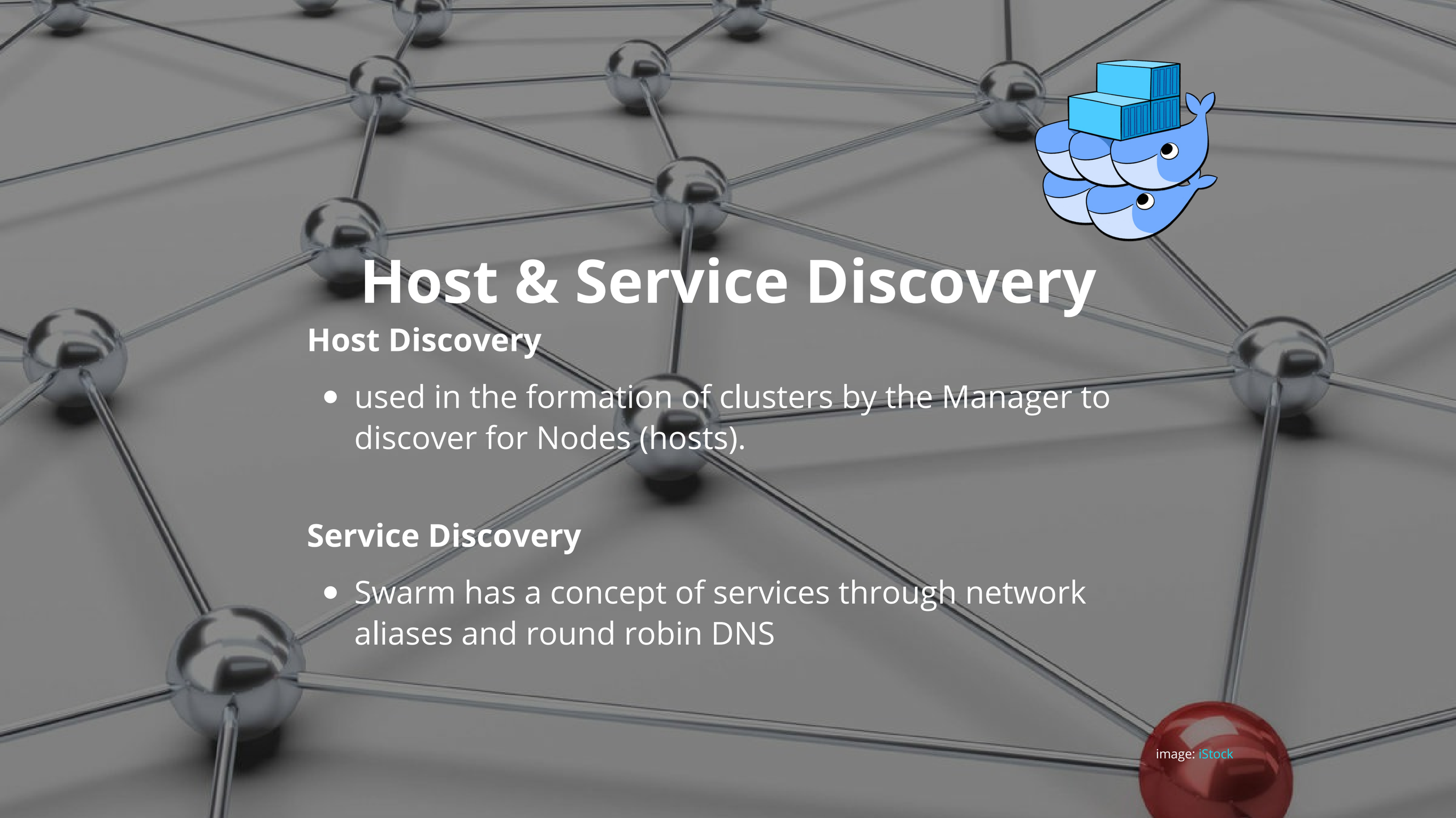

Docker
Client
Swarm Manager
Discovery Service
K/V
Database
Swarm Node
Swarm
Agent
Docker
Daemon



Support & Momentum

- ~3,000 commits by 12 core maintainers (130 contributors in all), and for Docker Compose, the numbers are approximately the same
- Swarm leverages the same API for Docker engine which over 23,000 commits and 1,350 contributors
- ~250 Docker meetups worldwide
- Announced as production-ready 5 months ago (Nov 2015)

image: digital trends



Host & Service Discovery

Host Discovery

- used in the formation of clusters by the Manager to discover for Nodes (hosts).

Service Discovery

- Swarm has a concept of services through network aliases and round robin DNS

image: iStock

Scheduling

- Swarm's scheduler is pluggable
- Swarm scheduling is a combination of strategies and filters/constraint:
  - **Strategies**
    - Random
    - Binpack
    - Spread*
  - **Filters**
    - **container constraints** (affinity, dependency, port) are defined as environment variables in the specification file
    - **node constraints** (health, constraint) must be specified when starting the docker daemon and define which nodes a container may be scheduled on.

image: pickywallpapers

Modularity & Extensibility

Ability to remove batteries is a strength for Swarm:

- Pluggable scheduler
- Pluggable network driver
- Pluggable distributed K/V store
- Docker container engine runtime-only

image: Alan Chia

Updates & Maintenance

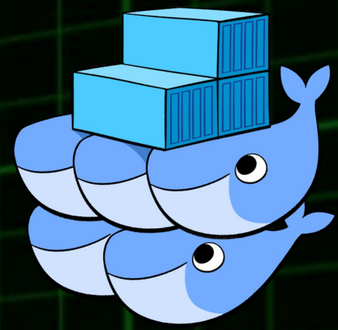
Nodes

- Manual swarm manager and agent updates

Applications

- No facilities for updating application

image: 123RF



Health Monitoring

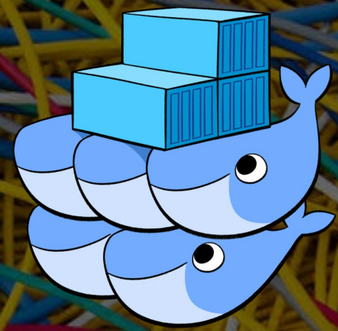
Nodes

- Swarm monitors the availability and resource usage of nodes within the cluster

Applications

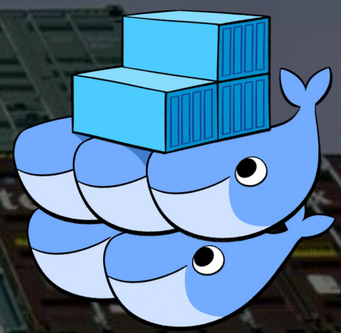
- No facilities for monitoring applications
- Use third-party software

Networking & Load-Balancing



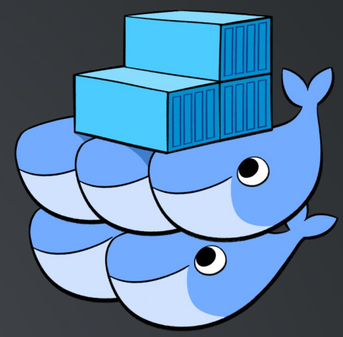
- Docker Swarm is fully compatible with Docker's networking
- Docker multi-host networking (requires a K/V store) provides for user-defined overlay networks that are micro-segmentable
  - uses a gossip protocol for quick convergence of neighbor table
  - facilitates container name resolution via embedded DNS server (previously via etc/hosts)
- You may bring your own network driver
- Load-balancing is new in 1.11.0
  - - no documentation available

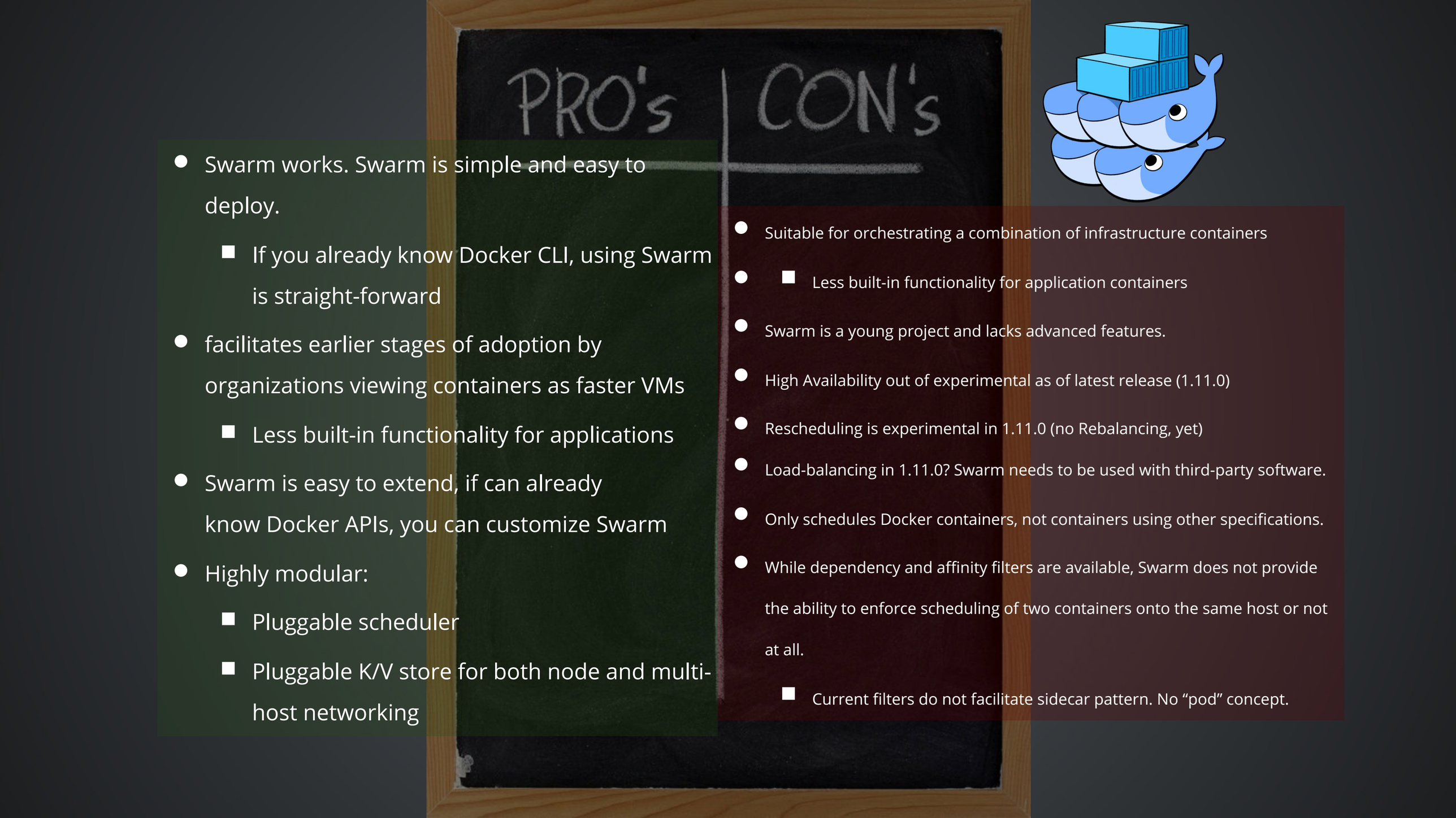
High Availability & Scale



- Managers may be deployed in a highly-available configuration (out of experimental in 1.11.0)
  - Active/Standby - an active manager and replicas configuration
- Rescheduling upon node failure (experimental in 1.11.0) and simple.
- Rebalancing is not available
  - Rescheduled containers lose access to volumes mounted on the former host.
    - use volume plugins like Flocker to avoid this
- Does not support multiple failure isolation regions or federation (although, with caveats, this is possible).

Scaling swarm to 1,000 AWS nodes and 50,000 containers





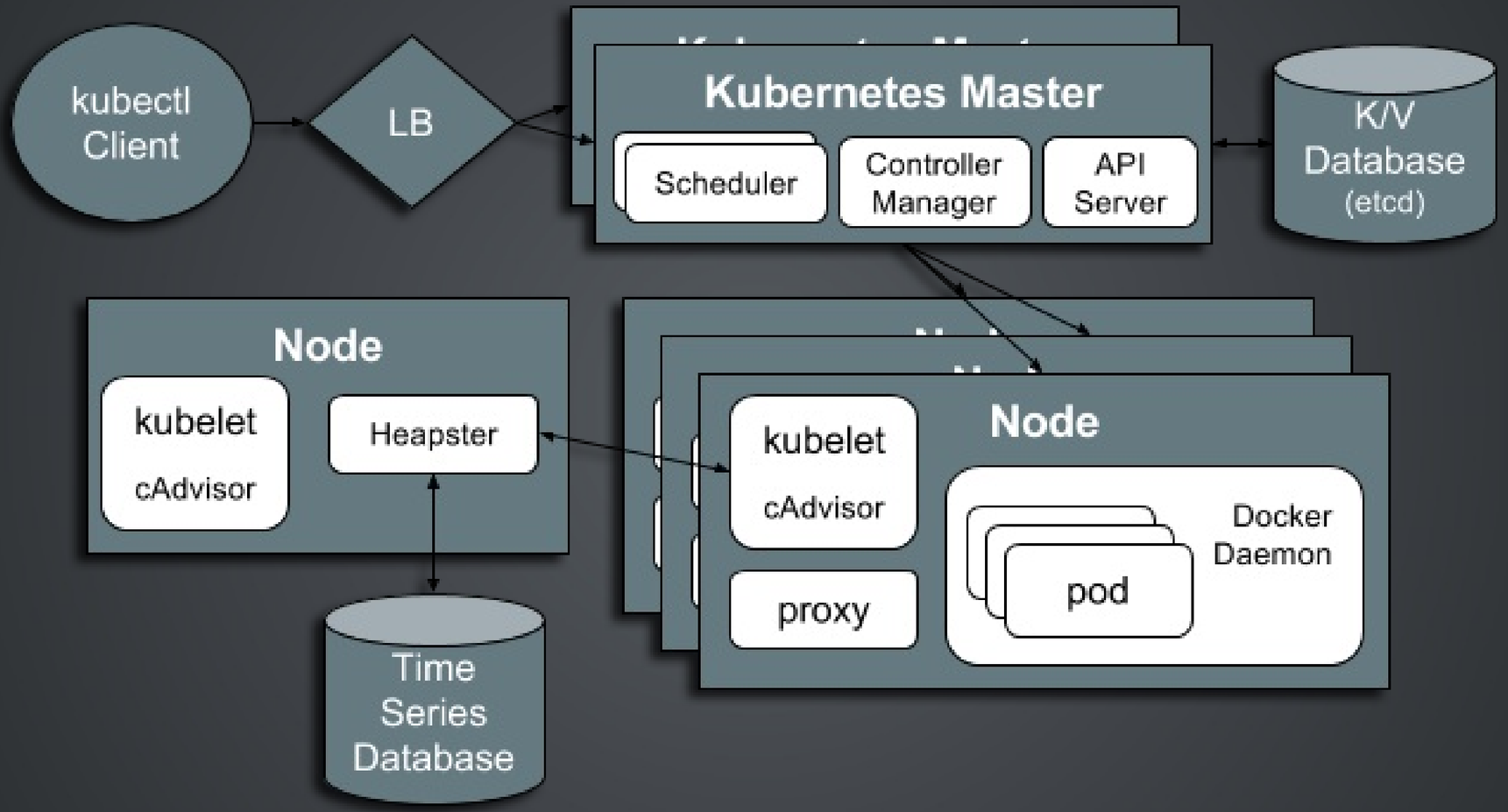
- Swarm works. Swarm is simple and easy to deploy.
  - If you already know Docker CLI, using Swarm is straight-forward
- facilitates earlier stages of adoption by organizations viewing containers as faster VMs
  - Less built-in functionality for applications
- Swarm is easy to extend, if can already know Docker APIs, you can customize Swarm
- Highly modular:
  - Pluggable scheduler
  - Pluggable K/V store for both node and multi-host networking

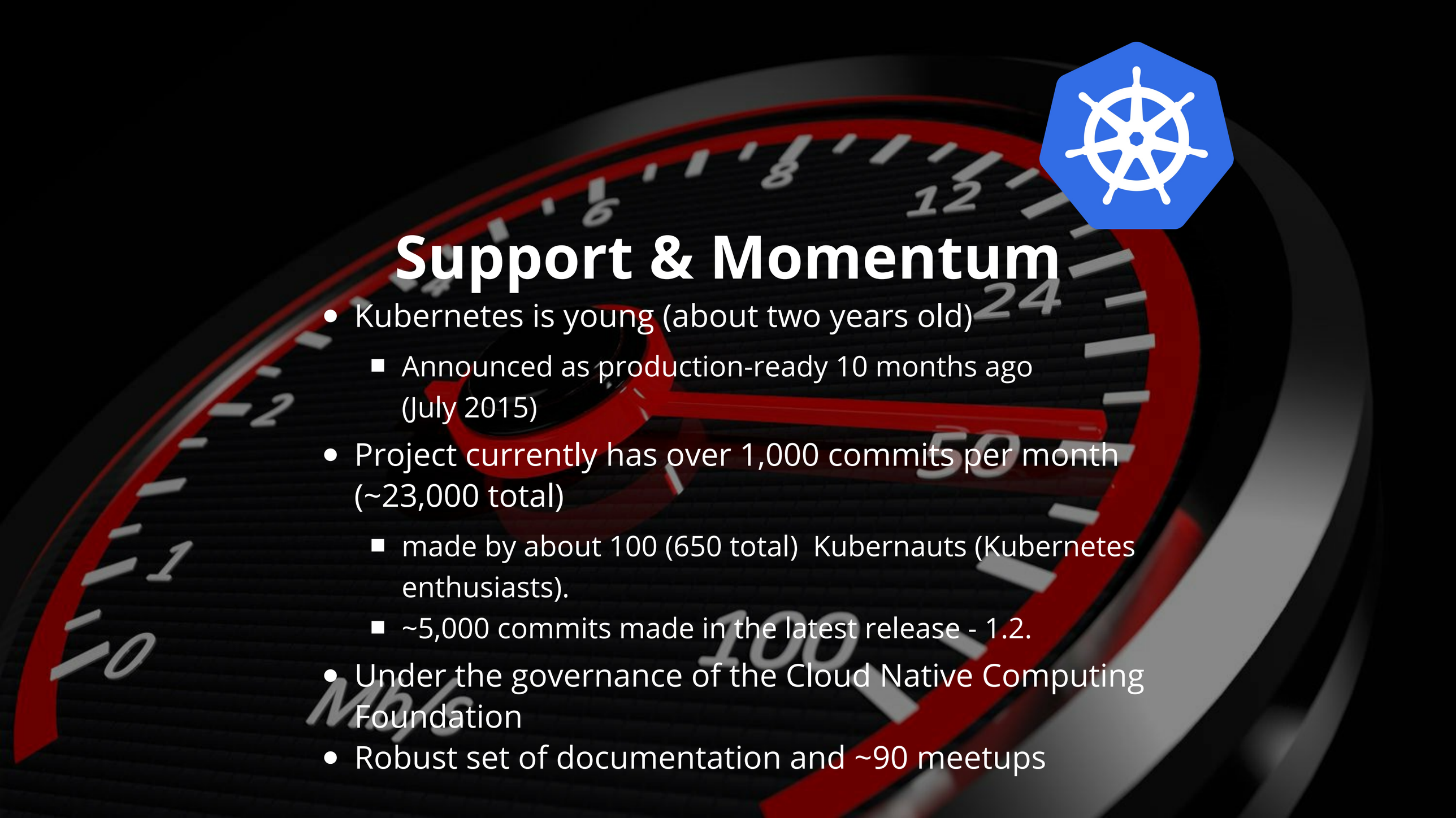
- Suitable for orchestrating a combination of infrastructure containers
- - Less built-in functionality for application containers
- Swarm is a young project and lacks advanced features.
- High Availability out of experimental as of latest release (1.11.0)
- Rescheduling is experimental in 1.11.0 (no Rebalancing, yet)
- Load-balancing in 1.11.0? Swarm needs to be used with third-party software.
- Only schedules Docker containers, not containers using other specifications.
- While dependency and affinity filters are available, Swarm does not provide the ability to enforce scheduling of two containers onto the same host or not at all.
  - Current filters do not facilitate sidecar pattern. No “pod” concept.


Kubernetes

Genesis & Purpose

- an opinionated framework for building distributed systems
  - or as its tagline states "an open source system for automating deployment, scaling, and operations of applications."
- Written in Golang, Kubernetes is lightweight, modular, portable, and extensible
- considered a third generation container orchestrator led by Google, Red Hat and others.
  - bakes in load-balancing, scale, volumes, deployments and secret management among other features.
- Declaratively, opinionated with many key features included


kubectl
Client
LB
Kubernetes Master
Scheduler
Controller
Manager
API
Server
K/V
Database
(etcd)
Node
kubelet
cAdvisor
Heapster
Time
Series
Database
kubelet
cAdvisor
proxy
Node
Docker
Daemon
pod



Support & Momentum

- Kubernetes is young (about two years old)
  - Announced as production-ready 10 months ago (July 2015)
- Project currently has over 1,000 commits per month (~23,000 total)
  - made by about 100 (650 total) Kubernauts (Kubernetes enthusiasts).
  - ~5,000 commits made in the latest release - 1.2.
- Under the governance of the Cloud Native Computing Foundation
- Robust set of documentation and ~90 meetups

Host & Service Discovery

Host Discovery

- by default, the node agent (kubelet) is configured to register itself with the master (API server)
  - automating the joining of new hosts to the cluster.

Service Discovery

Two primary modes of finding a Service

- environment variables
  - environment variables are used as a simple way of providing compatibility with Docker links-style networking
- DNS
  - when enabled is deployed as a cluster add-on

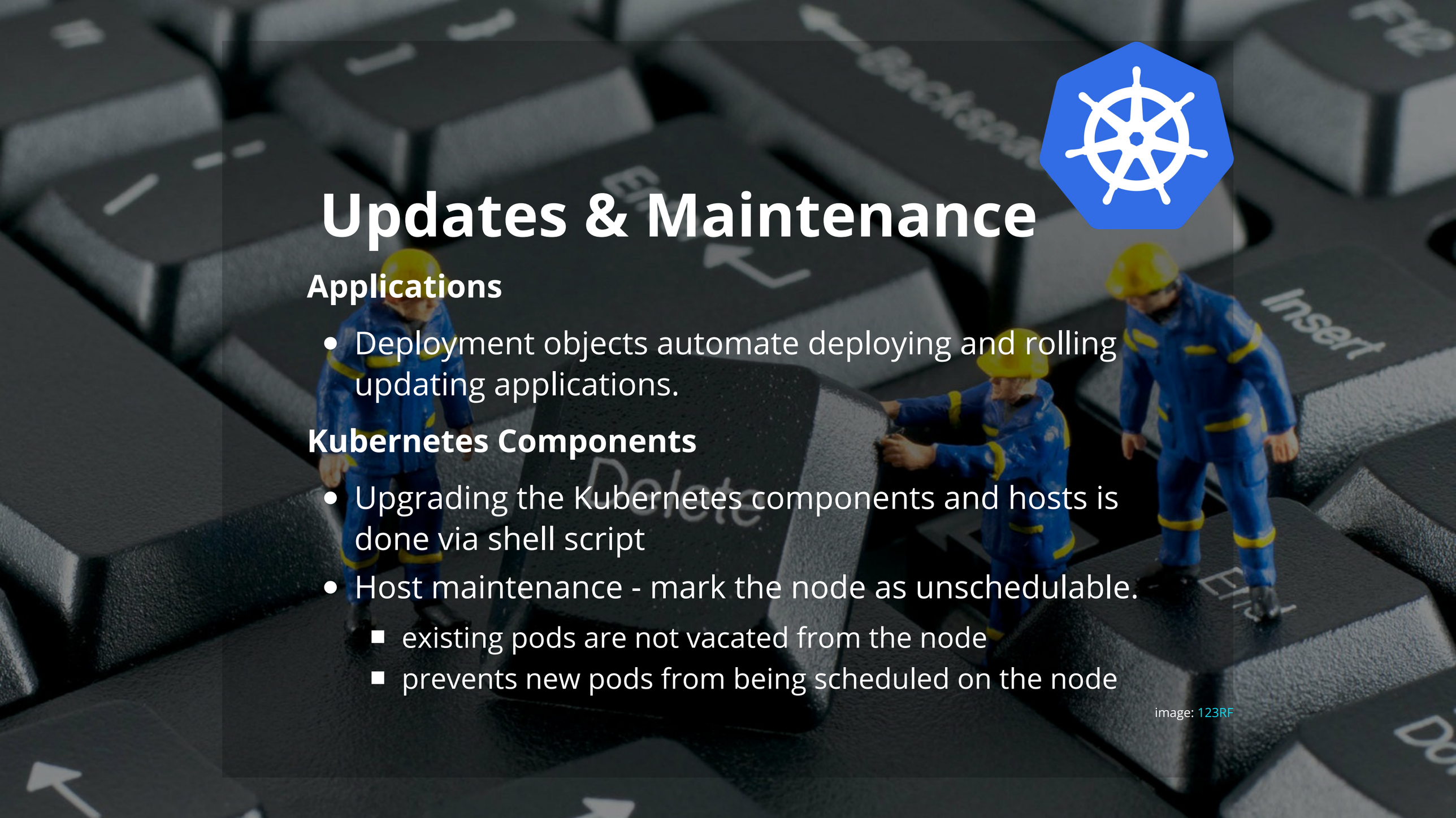
image: iStock

Scheduling

- By default, scheduling is handled by kube-scheduler.
- Pluggable
- Selection criteria used by kube-scheduler to identify the best-fit node is defined by policy:
  - Predicates (node resources and characteristics):
    - PodFitPorts , PodFitsResources, NoDiskConflict , MatchNodeSelector, HostName , ServiceAffinit, LabelsPresence
  - Priorities (weighted strategies used to identify “best fit” node):
    - LeastRequestedPriority, BalancedResourceAllocation, ServiceSpreadingPriority, EqualPriority

Modularity & Extensibility

- One of Kubernetes strengths its pluggable architecture
- Choice of:
  - database for service discovery or network driver
  - container runtime
    - users may choose to run Docker with Rocket containers
- Cluster add-ons
  - optional system components that implement a cluster feature (e.g. DNS, logging, etc.)
  - shipped with the Kubernetes binaries and are considered an inherent part of the Kubernetes clusters



Updates & Maintenance

Applications

- Deployment objects automate deploying and rolling updating applications.

Kubernetes Components

- Upgrading the Kubernetes components and hosts is done via shell script
- Host maintenance - mark the node as unschedulable.
  - existing pods are not vacated from the node
  - prevents new pods from being scheduled on the node

image: 123RF

Health Monitoring

Nodes

- Failures - actively monitors the health of nodes within the cluster
  - via Node Controller
- Resources - usage monitoring leverages a combination of open source components:
  - cAdvisor, Heapster, InfluxDB, Grafana

Applications

- three types of user-defined application health-checks and uses the Kubelet agent as the the health check monitor
  - HTTP Health Checks, Container Exec, TCP Socket

Cluster-level Logging

- collect logs which persist beyond the lifetime of the pod’s container images or the lifetime of the pod or even cluster
  - standard output and standard error output of each container can be ingested using a

Networking & Load-Balancing

...enter the Pod

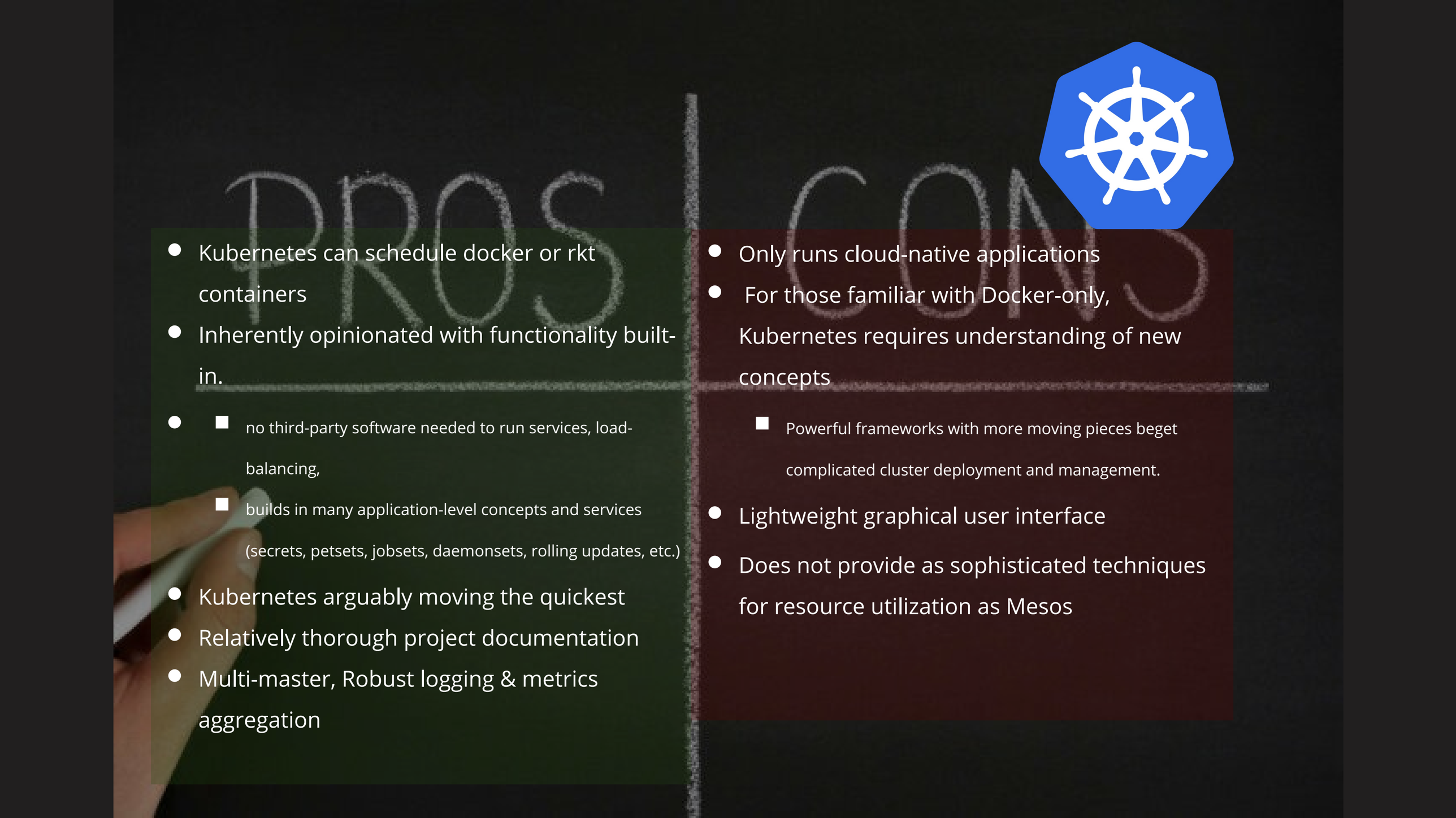
- atomic unit of scheduling
- flat networking with each pod receiving an IP address
- no NAT required, port conflicts localized
- intra-pod communication via localhost

Load-Balancing

- Services provide inherent load-balancing via kube-proxy:
  - runs on each node of a Kubernetes cluster
  - reflects services as defined in the Kubernetes API
  - supports simple TCP/UDP forwarding and round-robin and Docker-links-based service IP:PORT mapping.

High Availability & Scale

- Each master component may be deployed in a highly-available configuration.
  - Active/Standby configuration
- In terms of scale, v1.2 brings support for 1,000 node clusters and a step toward fully-federated clusters (Ubernetes)
- Application-level auto-scaling is supported within Kubernetes via Replication Controllers



- Kubernetes can schedule docker or rkt containers
- Inherently opinionated with functionality built-in.
  - no third-party software needed to run services, load-balancing,
  - builds in many application-level concepts and services (secrets, petsets, jobsets, daemonsets, rolling updates, etc.)
- Kubernetes arguably moving the quickest
- Relatively thorough project documentation
- Multi-master, Robust logging & metrics aggregation

- Only runs cloud-native applications
- For those familiar with Docker-only, Kubernetes requires understanding of new concepts
  - Powerful frameworks with more moving pieces beget complicated cluster deployment and management.
- Lightweight graphical user interface
- Does not provide as sophisticated techniques for resource utilization as Mesos

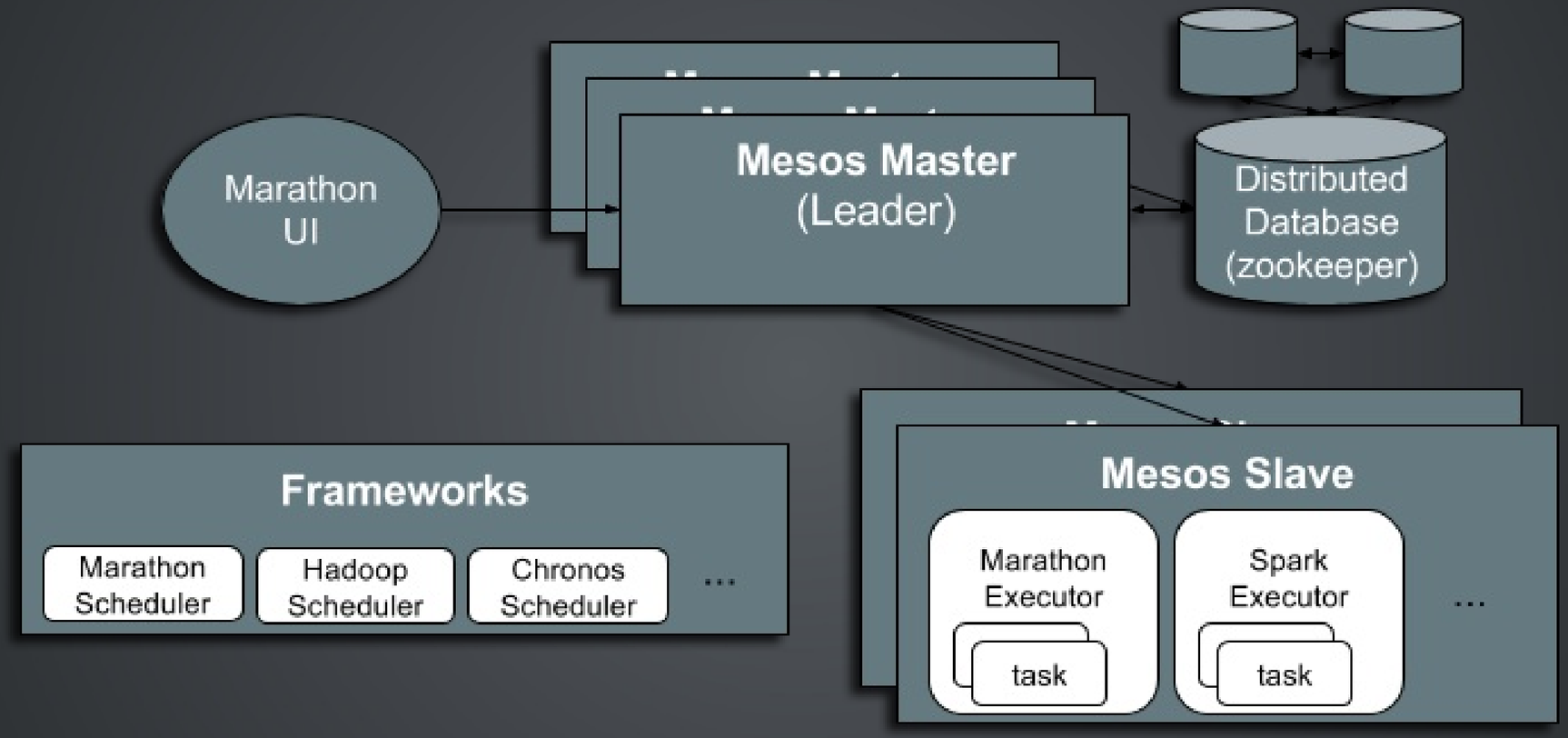
Mesos

+

Marathon

Genesis & Purpose

- Mesos is a distributed systems kernel
  - stitches together many different machines into a logical computer
- Mesos has been around the longest (launched in 2009)
  - and is arguably the most stable, with highest (proven) scale currently
- Mess is written in C++
  - with Java, Python and C++ APIs
- Frameworks
  - Marathon is one of a number of frameworks (Chronos and Aurora other examples) that may be run on top of Mesos
  - Frameworks have a scheduler and executor. Schedulers get resource offers. Executors run tasks.
  - Marathon is written in Scala

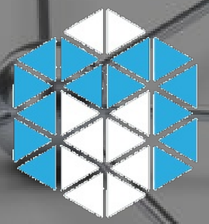

Marathon
UI
Mesos Master
(Leader)
Distributed
Database
(zookeeper)
Frameworks
Marathon
Scheduler
Hadoop
Scheduler
Chronos
Scheduler
...
Mesos Slave
Marathon
Executor
task
Spark
Executor
task
...

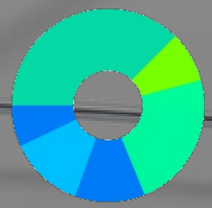
Support & Momentum

- MesosCon 2015 in Seattle had 700 attendees
  - up from 262 attendees in 2014
- 78 contributors in the last year

- Under the governance of Apache Foundation
- Used by Twitter, AirBnb, eBay, Apple, Cisco, Yodle

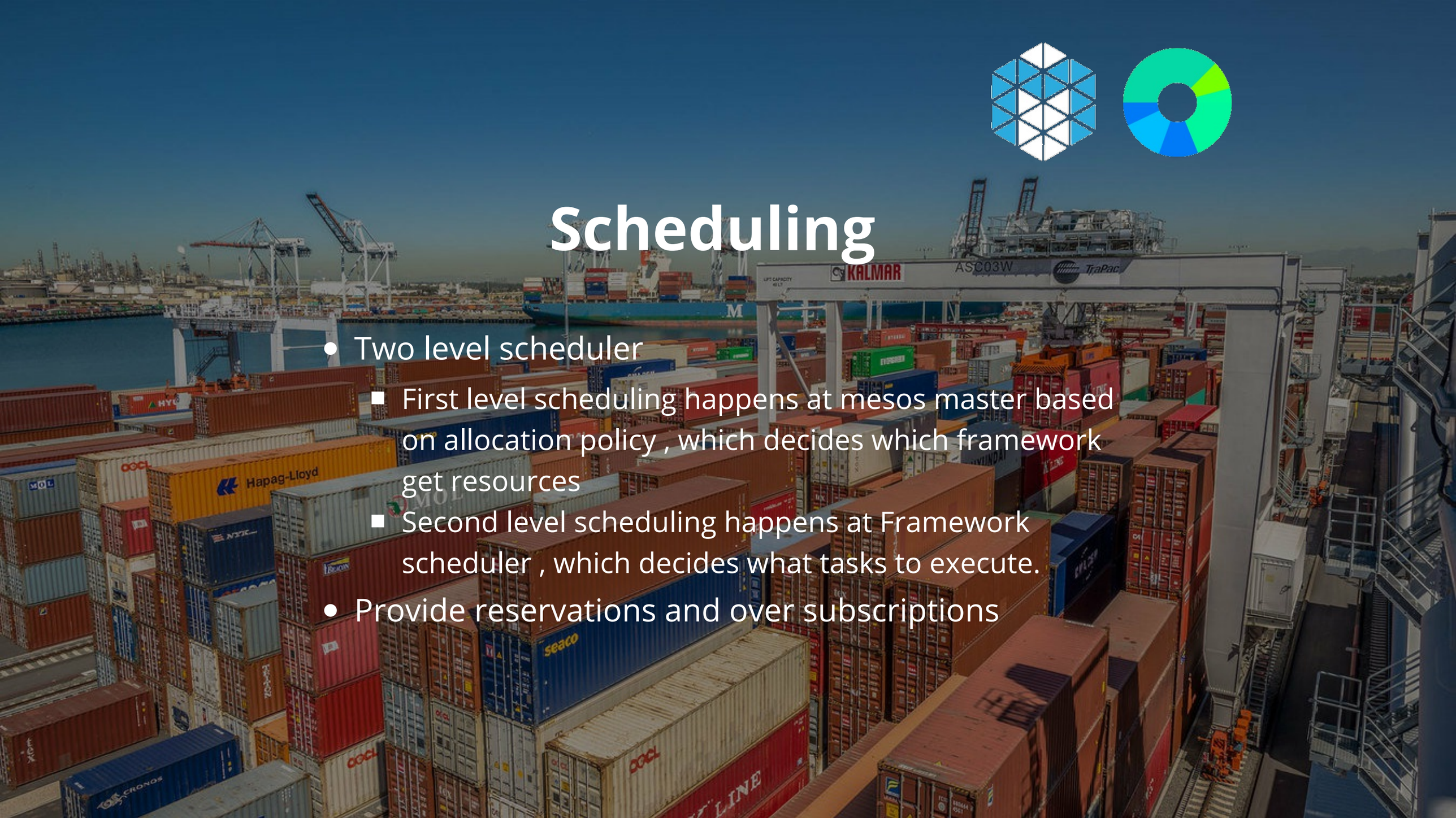
Host & Service Discovery





- Mesos-DNS generates an SRV record for each Mesos task
  - including Marathon application instances
- Marathon will ensure that all dynamically assigned service ports are unique
- Mesos-DNS is particularly useful when:
  - apps are launched through multiple frameworks (not just Marathon)
  - you are using an IP-per-container solution like Project Calico
  - you use random host port assignments in Marathon

image: iStock



Scheduling

- Two level scheduler
  - First level scheduling happens at mesos master based on allocation policy , which decides which framework get resources
  - Second level scheduling happens at Framework scheduler , which decides what tasks to execute.
- Provide reservations and over subscriptions

Modularity & Extensibility

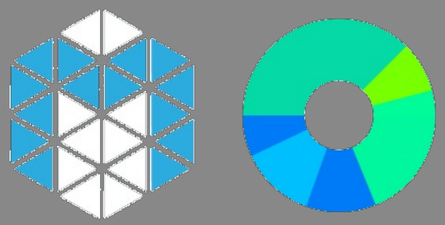
Frameworks

- multiple available
- may run multiple frameworks

Modules

- has three types of Modules
  - Replacement, Hook, Anonymous

Updates & Maintenance



Nodes

- Mesos has maintenance mode

Applications

- Marathon can be instructed to deploy containers based on that component using a blue/green strategy
  - where old and new versions co-exist for a time.

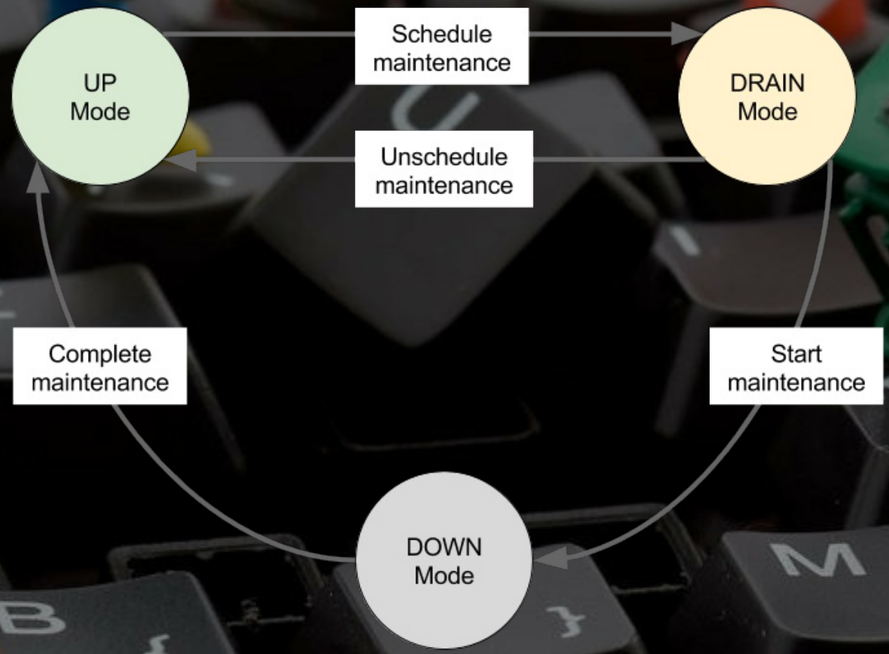
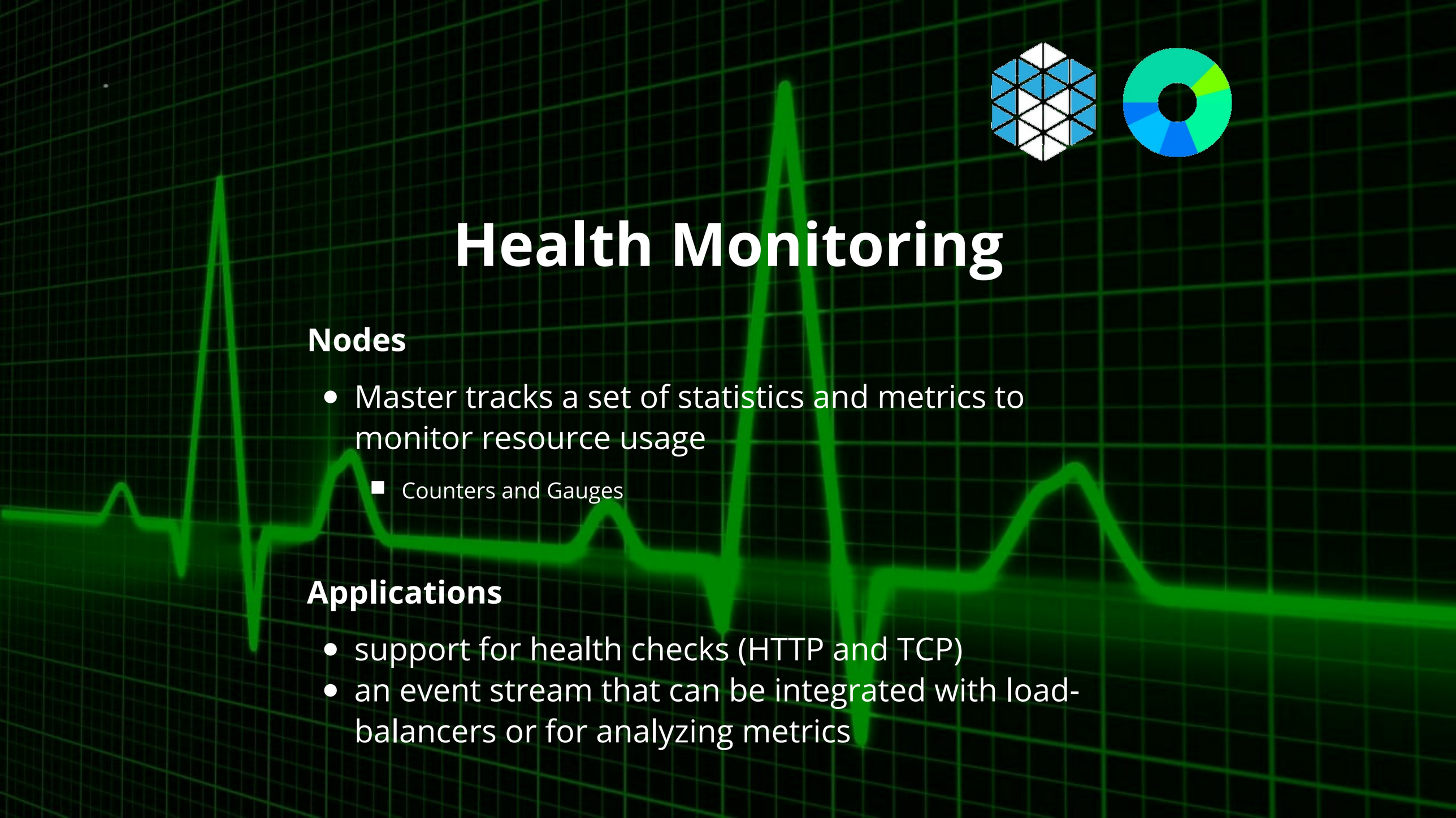


image: 123RF



Health Monitoring

Nodes

- Master tracks a set of statistics and metrics to monitor resource usage
  - Counters and Gauges

Applications

- support for health checks (HTTP and TCP)
- an event stream that can be integrated with load-balancers or for analyzing metrics

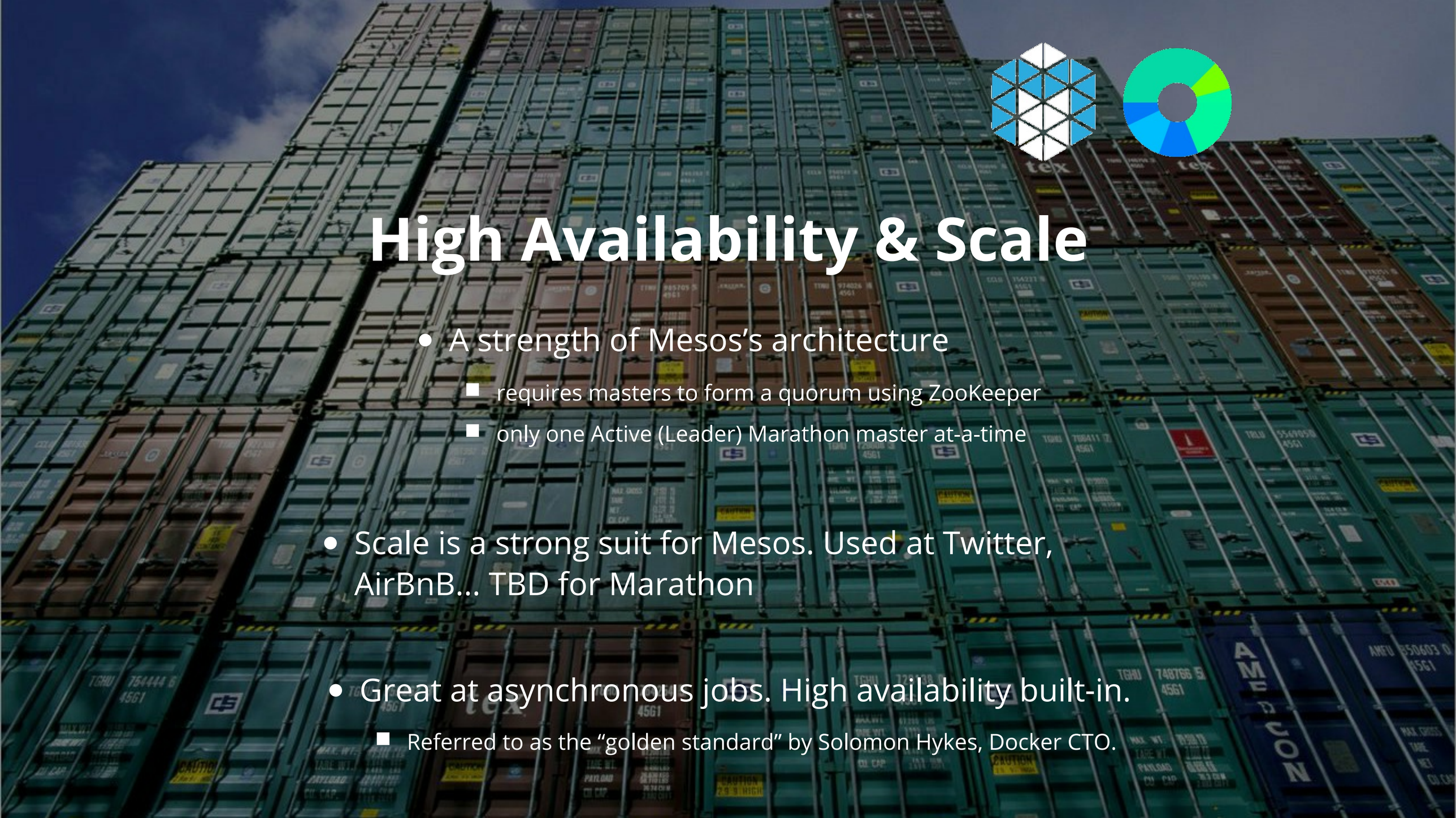
Networking & Load-Balancing

Networking

- An IP per Container
  - No longer share the node's IP
- Helps remove port conflicts
- Enables 3rd party network drivers

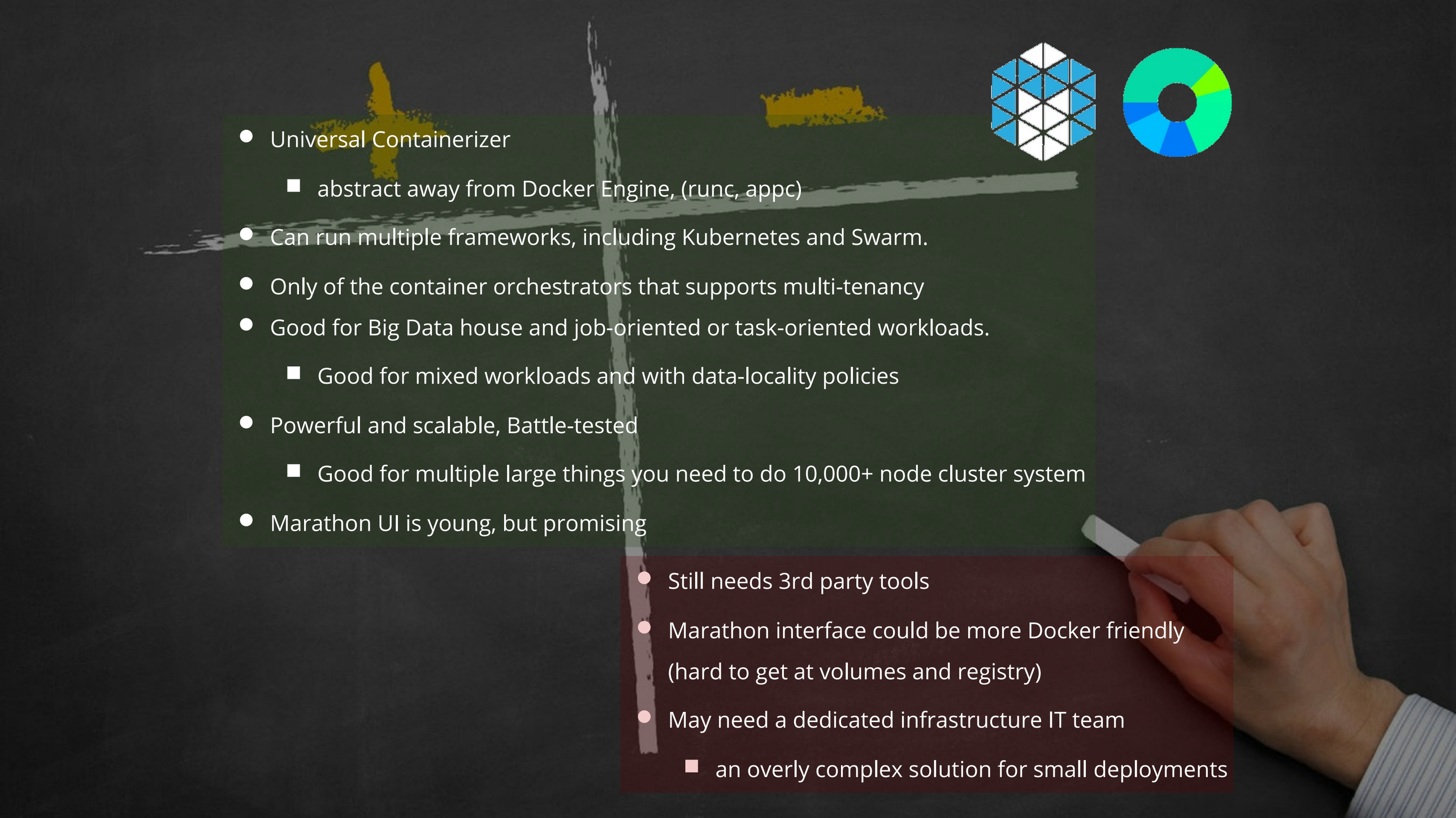
Load-Balancing

- Marathon offers two TCP/HTTP proxy.
  - A simple shell script and a more complex one called marathon-lb that has more features.
  - Pluggable (e.g. Traefic for load-balancing)



High Availability & Scale

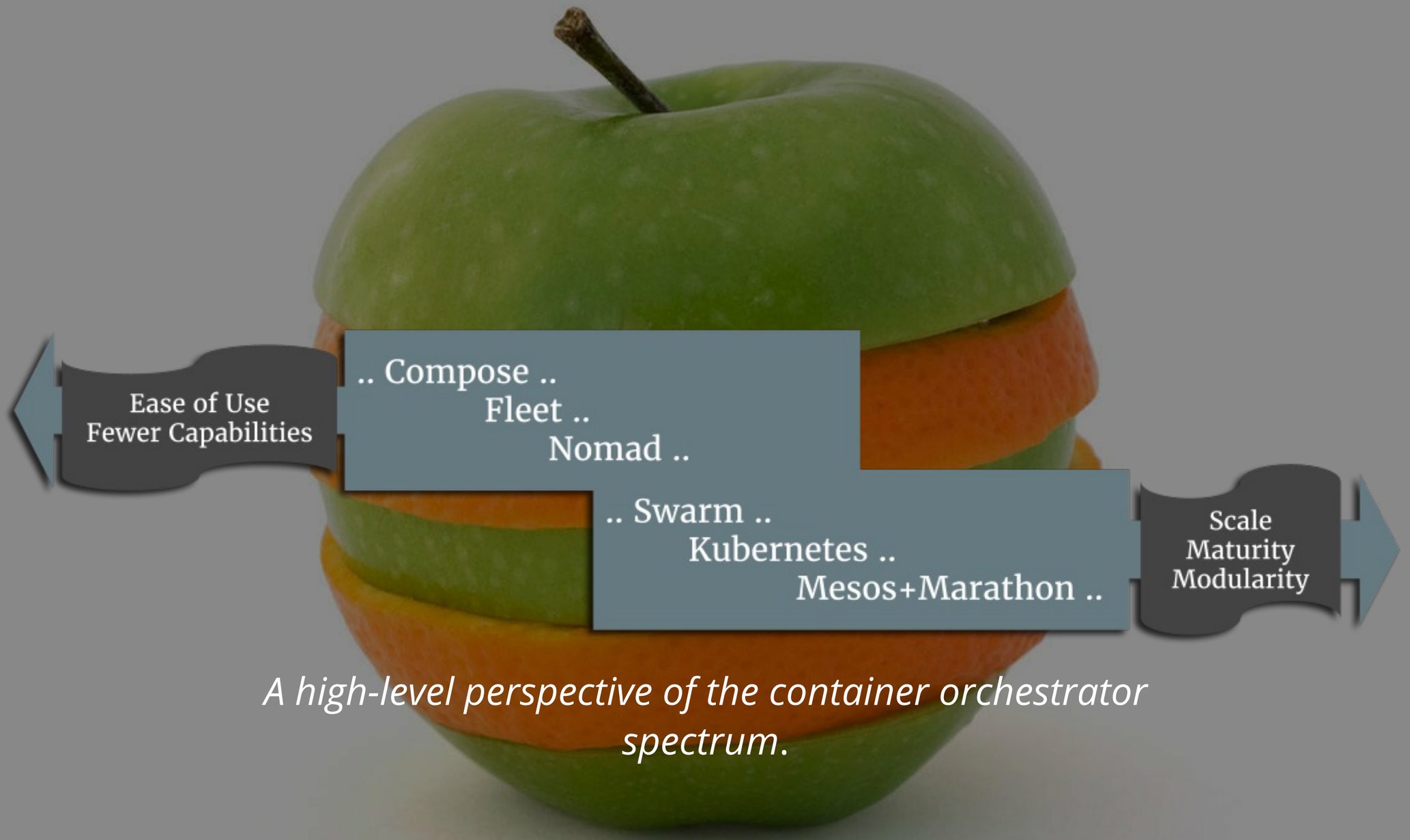
- A strength of Mesos's architecture
  - requires masters to form a quorum using ZooKeeper
  - only one Active (Leader) Marathon master at-a-time
- Scale is a strong suit for Mesos. Used at Twitter, AirBnB... TBD for Marathon
- Great at asynchronous jobs. High availability built-in.
  - Referred to as the "golden standard" by Solomon Hykes, Docker CTO.



- Universal Containerizer
  - abstract away from Docker Engine, (runc, appc)
- Can run multiple frameworks, including Kubernetes and Swarm.
- Only of the container orchestrators that supports multi-tenancy
- Good for Big Data house and job-oriented or task-oriented workloads.
  - Good for mixed workloads and with data-locality policies
- Powerful and scalable, Battle-tested
  - Good for multiple large things you need to do 10,000+ node cluster system
- Marathon UI is young, but promising

- Still needs 3rd party tools
- Marathon interface could be more Docker friendly (hard to get at volumes and registry)
- May need a dedicated infrastructure IT team
  - an overly complex solution for small deployments

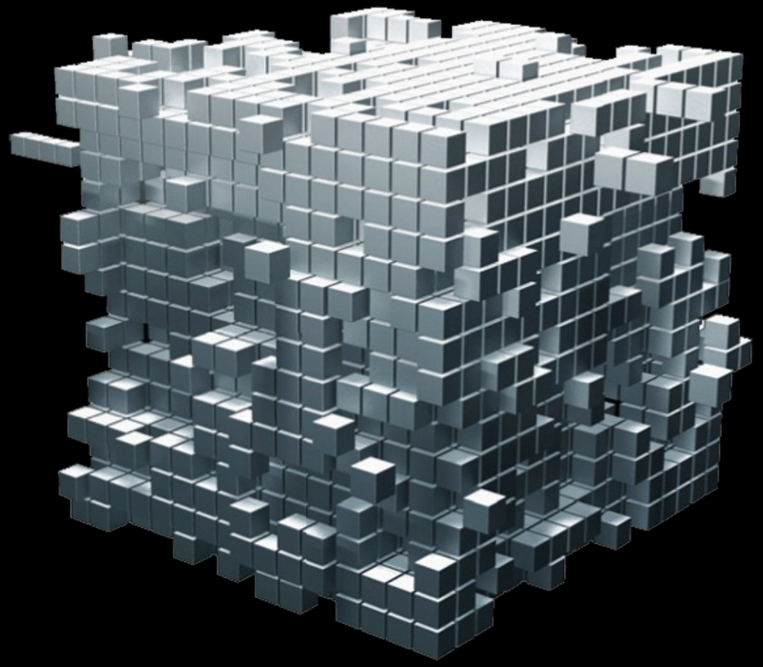
Summary



A high-level perspective of the container orchestrator spectrum.

Lee Calcote

Thank you. Questions?



@lcalcote

blog.gingergeek.com

linkedin.com/in/leecalcote

lee@calcotestudios.com