
DOCKER AND KUBERNETES
dhananjayan

Day 4

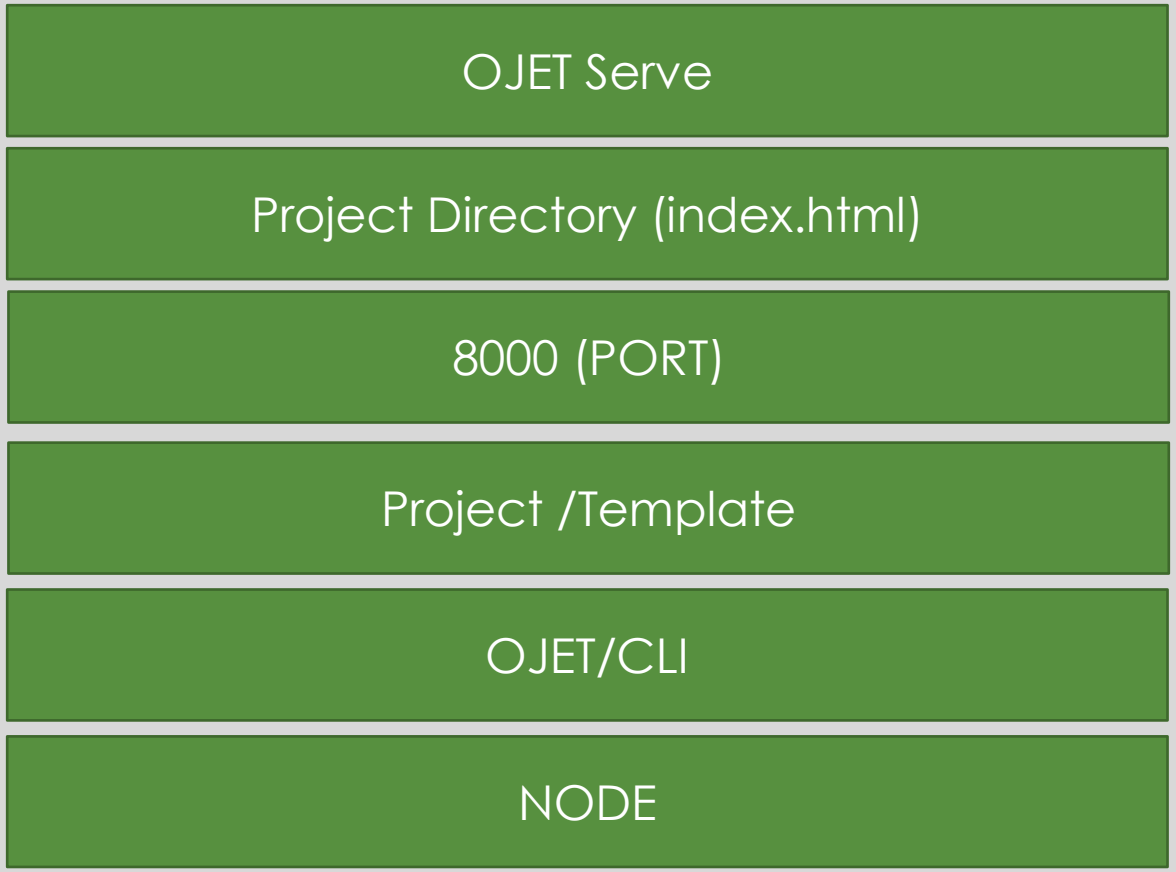
- Trouble shooting (Log Policies)
- UI OJET Container…
- Container policies (Transient)
- Release management
- Release images
- Kubernetes Architecture
- Kubernetes Installation
- Kubernetes Pods
- Kubernetes Objects
- Service management (k8s)

Log configuration properties..

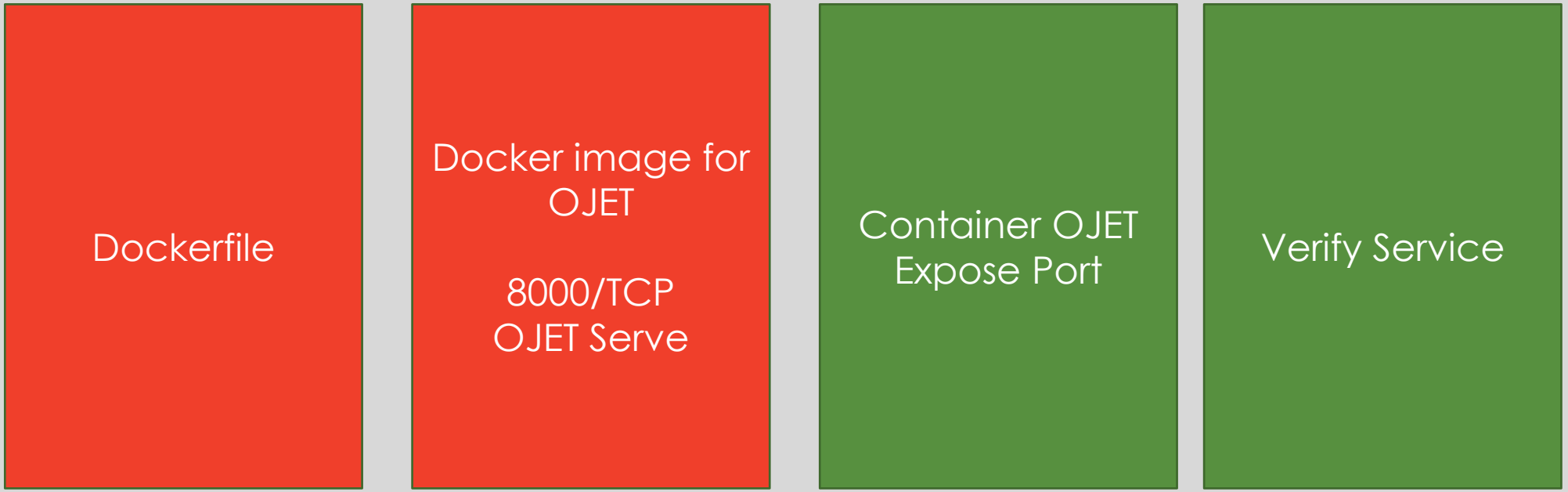
# Log Files	Log Buffer Size	Log Mode	Log Mode
1 log file	4M	**Blocking** _once log file reaches log size – no more logs written	JSON-FILE FLUENTD (EFK) SYSLOG SPLUNK TextLogs Log4J Awslogs/gclogs None
32767 log files	K to G to T	Non-blocking : -- cyclic logs , logs are rewritten , new logs as assurance	JSON-FILE
/var/lib/docker	dockerd (memory+swap)	blocking	JSON-FILE
Custom → 10 Logs	Size→ 1K	Non-blocking	JSON-file

UI

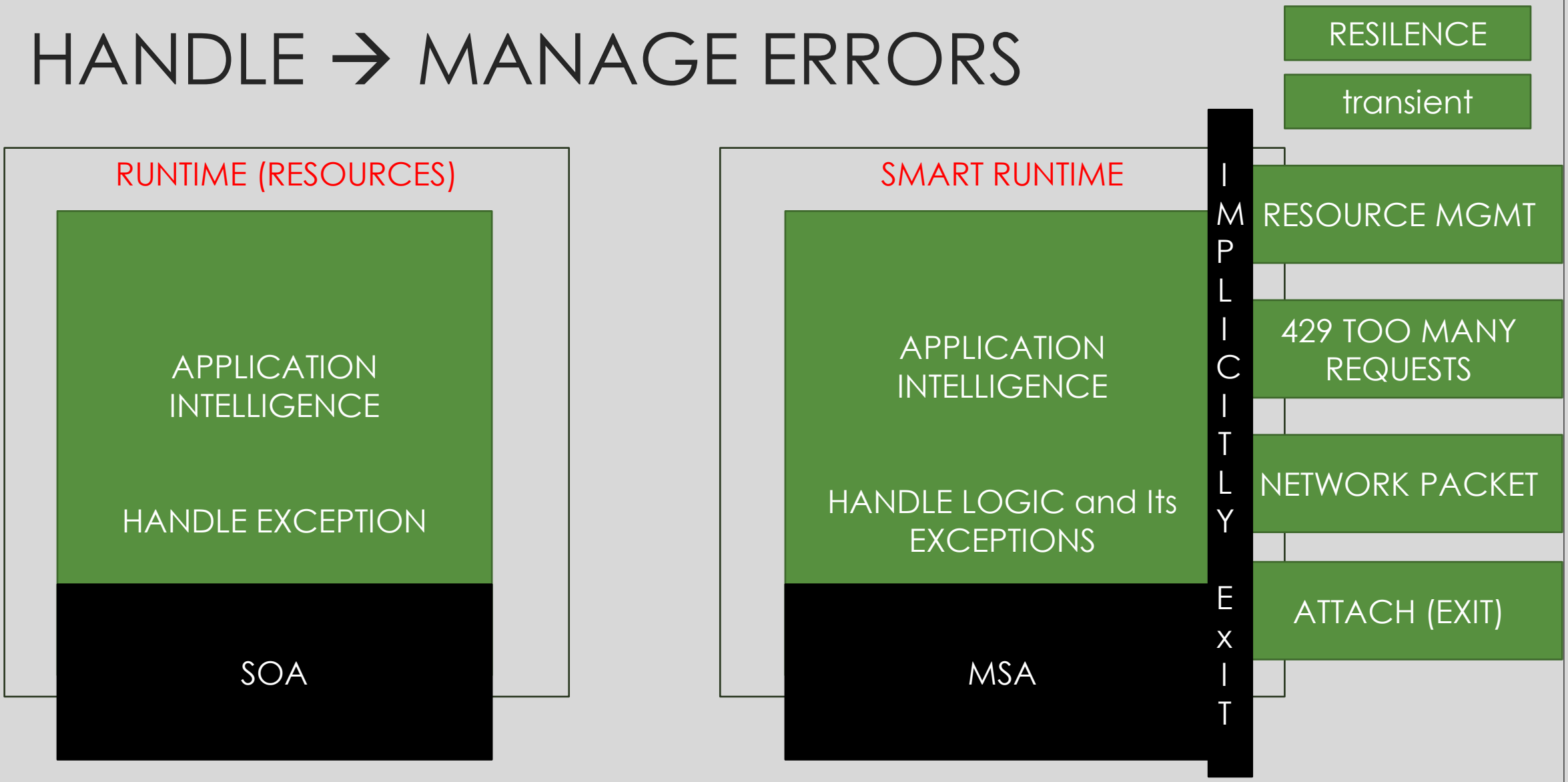
- Open responsive design
- OJET – HTML CSS , Require.JS , Knockout,jS (data binding)
- Node JS



Implement UI



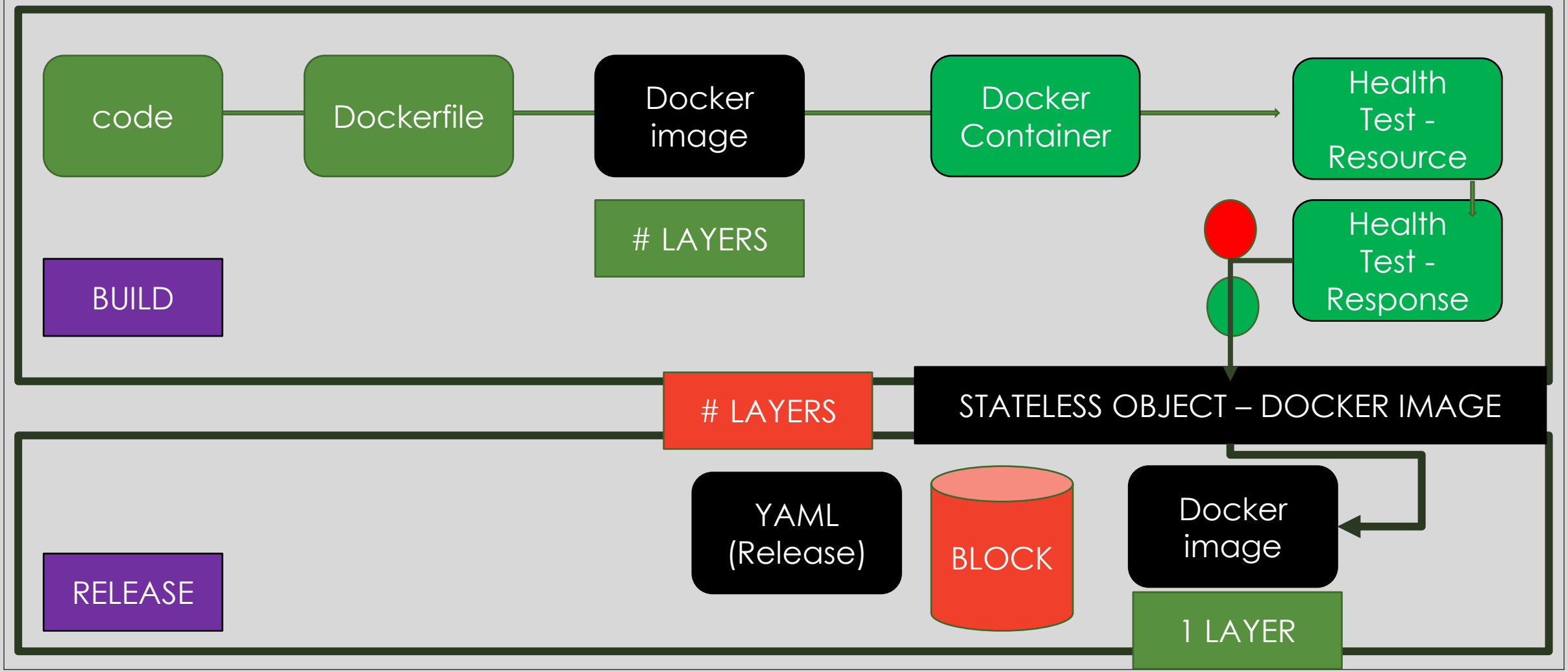
HANDLE → MANAGE ERRORS



CONTAINER POLICY → RESTART POLICY

#NEVER (NO)	# ALWAYS	# ON-FAILURE:X
No restart for Implicity exit	Indefinite restarts when there is implicit exit	Restarting on failure of the application, X threshold of the restart

Release management



Release automation

CLI (manual)
End to end

Network (Bridge)

Container 1

Image1

Ports, Configurations, Settings

Container 2

Image1

Ports, Configurations, Settings

YAML (declarative)
Create .Destruct (release)

Key: value

Key : {json}, key: |
Key: [json], key: -

Indendation

#docker-compose

Yaml file

```
version: 3
services
  database:
    image: newmysql:1
   web:
     image: nginx
     ports:
     - “2000:80”
```

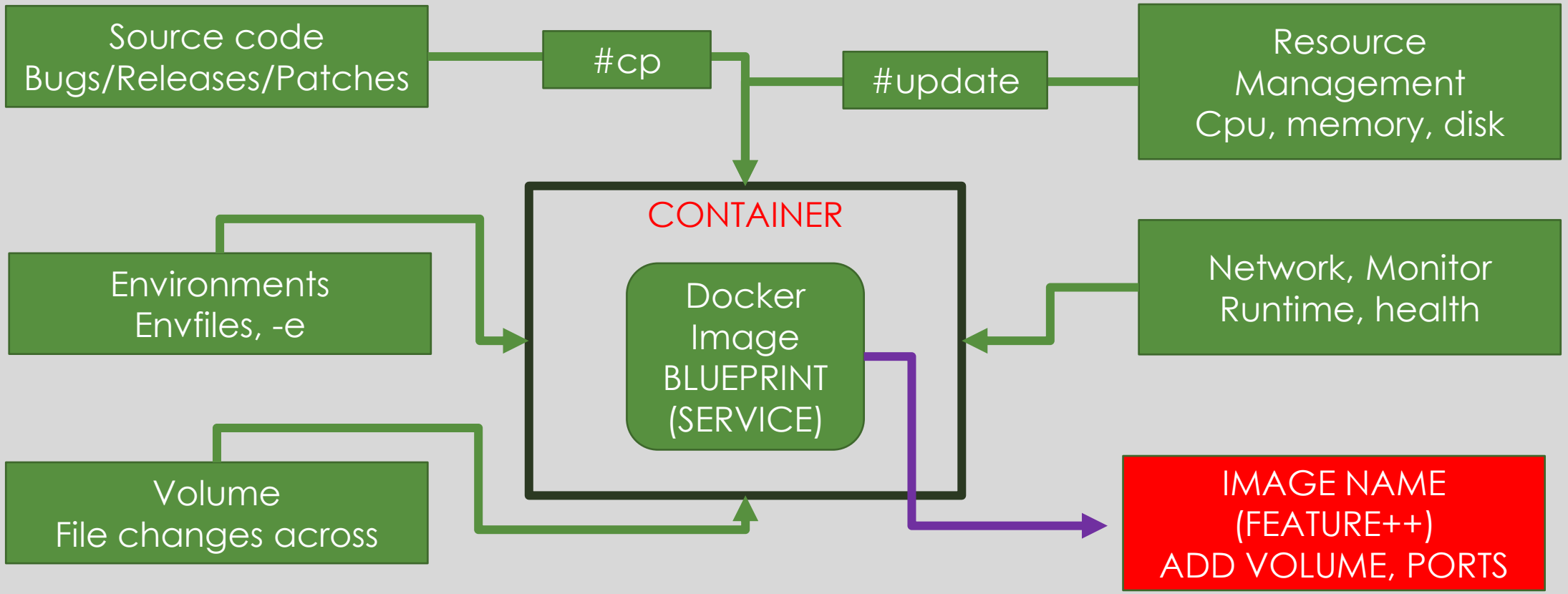
Compose version

Meta data for container

Meta data for container 2

Static Port forwarding of container

Minimizing changes…



Environment for K8s

Use case(s)

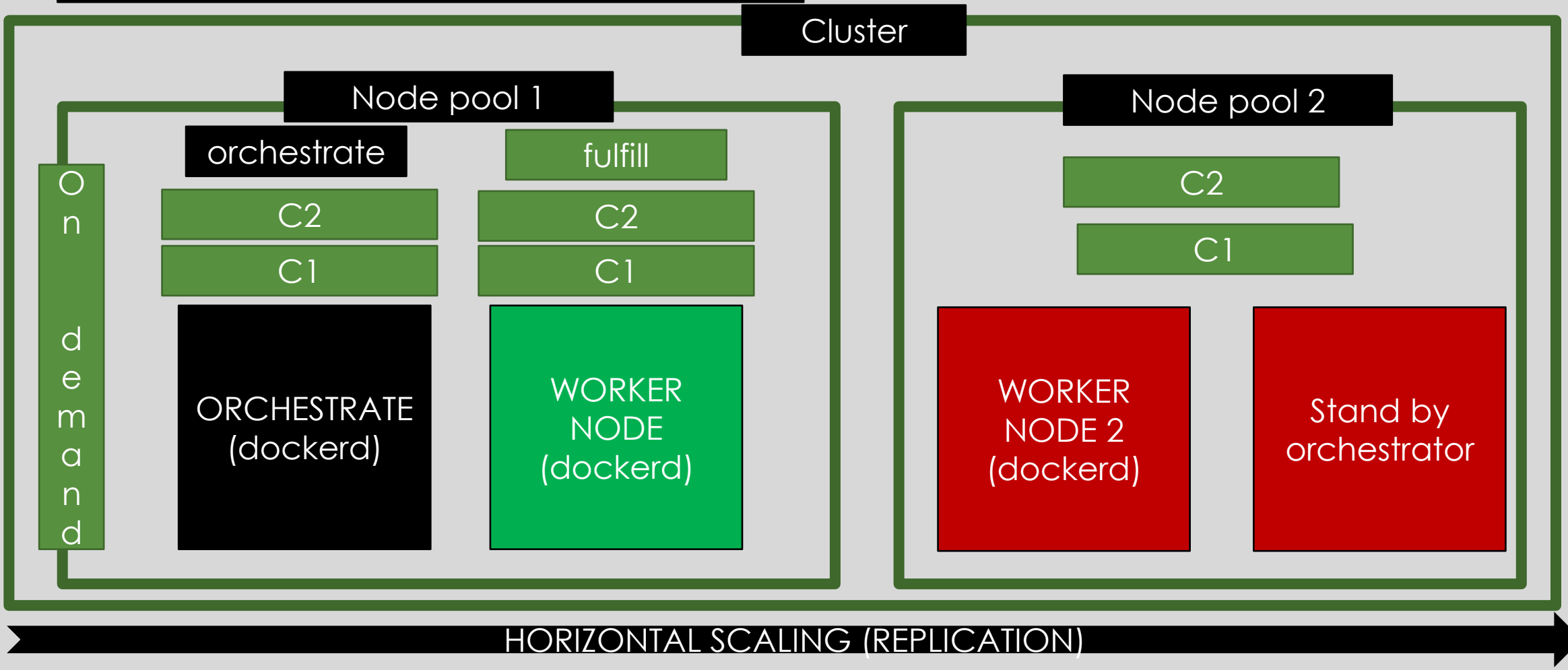
Why to use Docker	Why to use K8s
+ Applications are Portable, Runtime are not portable. (Runtimes –Application changes) [PORTABILITY] + Performance of Application + Option to Scale as a Service	+ Orchestrate Services as Containers (heal) + Horizontal (machines), Vertical (Containers) + Rollout and Release (change mgmt.) +Any Container runtime

Horizontal scaling

Node=machine

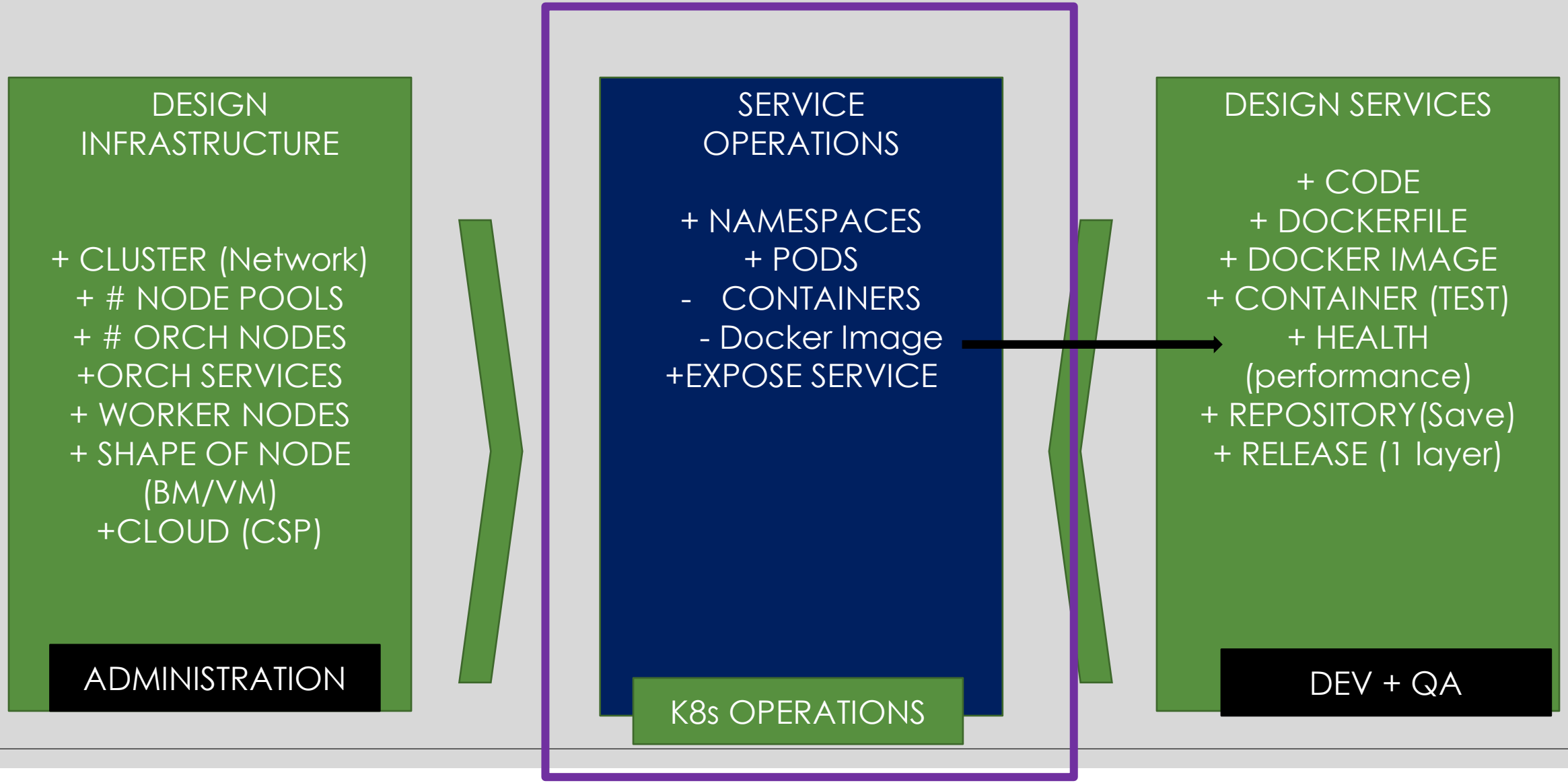
VM/BM = $

Monitor,Load Balance, log and fulfill

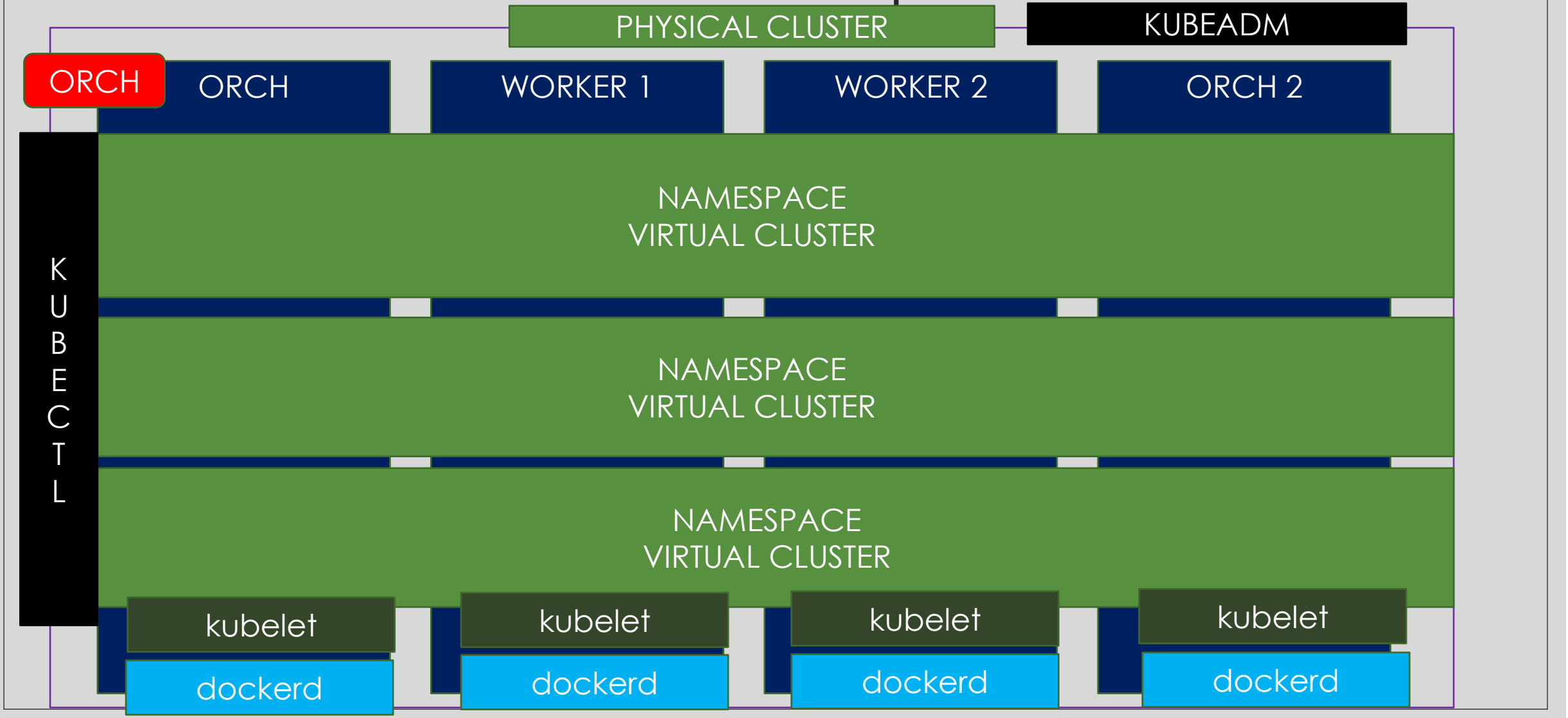


HORIZONTAL SCALING (REPLICATION)

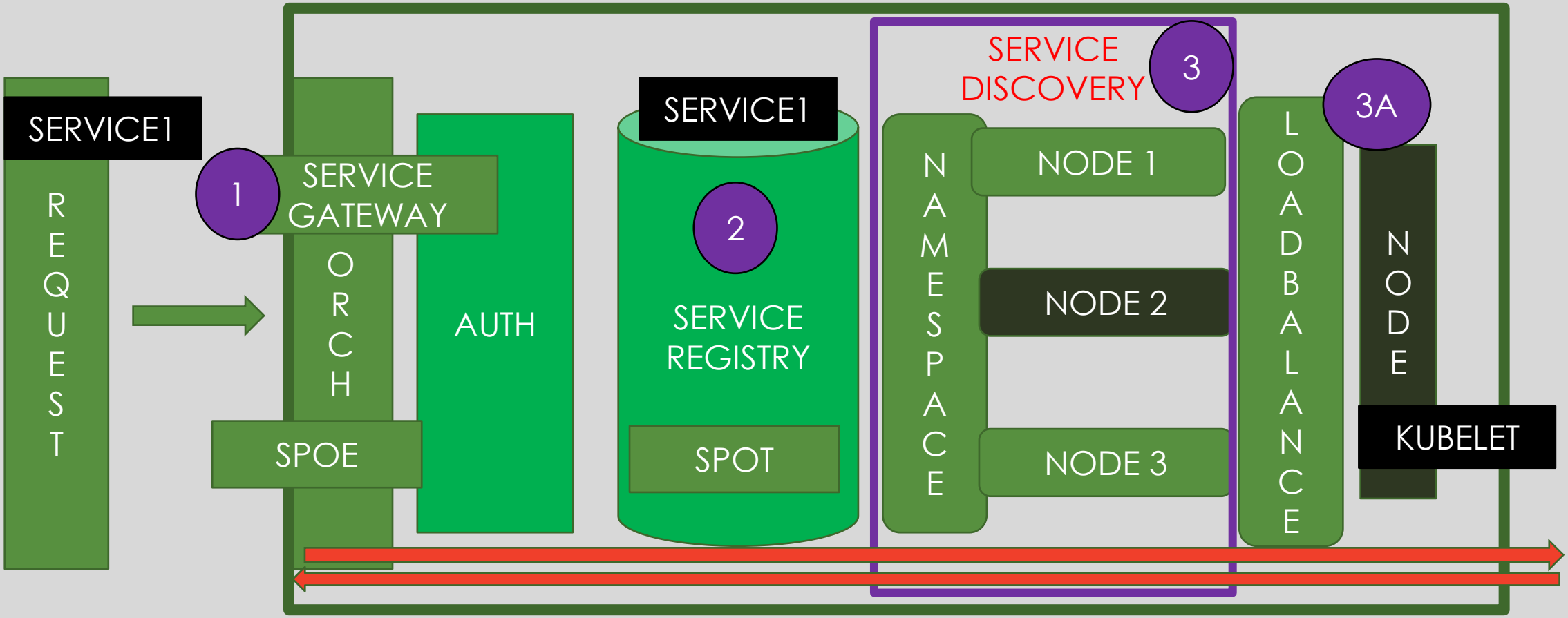
3 roles


DESIGN
INFRASTRUCTURE
+ CLUSTER (Network)
+ # NODE POOLS
+ # ORCH NODES
+ORCH SERVICES
+ WORKER NODES
+ SHAPE OF NODE
(BM/VM)
+CLOUD (CSP)
ADMINISTRATION
SERVICE
OPERATIONS
+ NAMESPACES
+ PODS
- CONTAINERS
- Docker Image
+EXPOSE SERVICE
K8s OPERATIONS
DESIGN SERVICES
+ CODE
+ DOCKERFILE
+ DOCKER IMAGE
+ CONTAINER (TEST)
+ HEALTH
(performance)
+ REPOSITORY(Save)
+ RELEASE (1 layer)
DEV + QA

Virtual cluster → Namespace



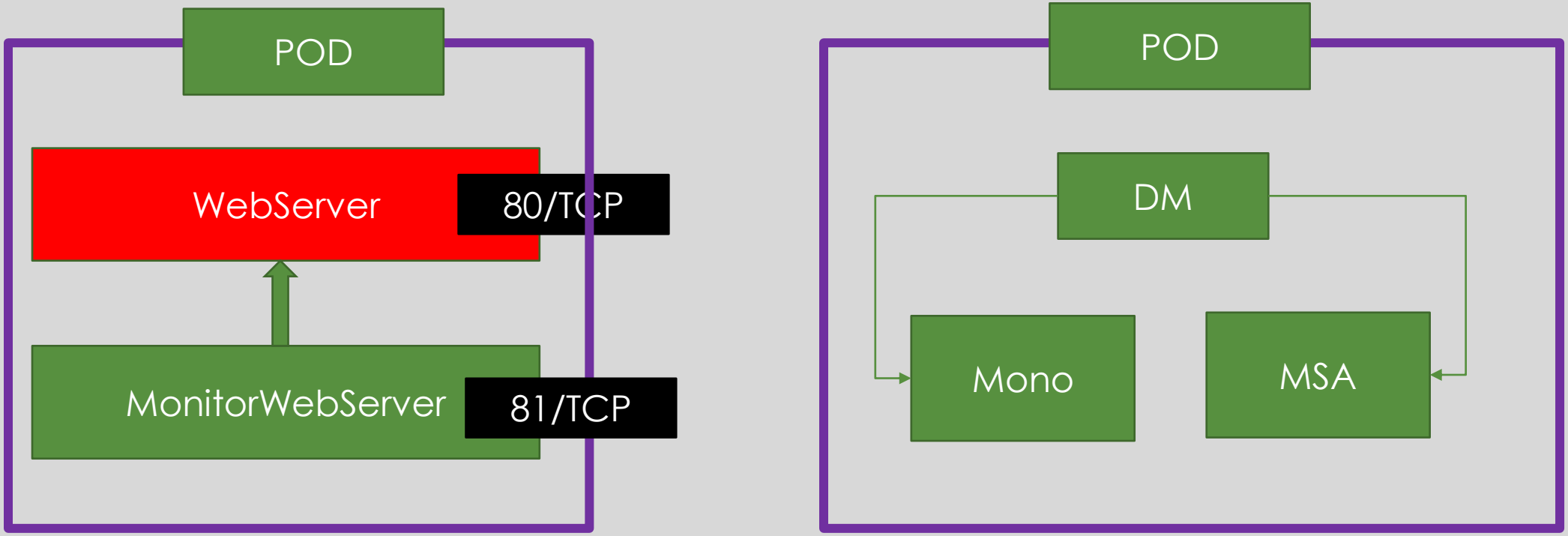
Orchestrator...



Orchestrator tools…

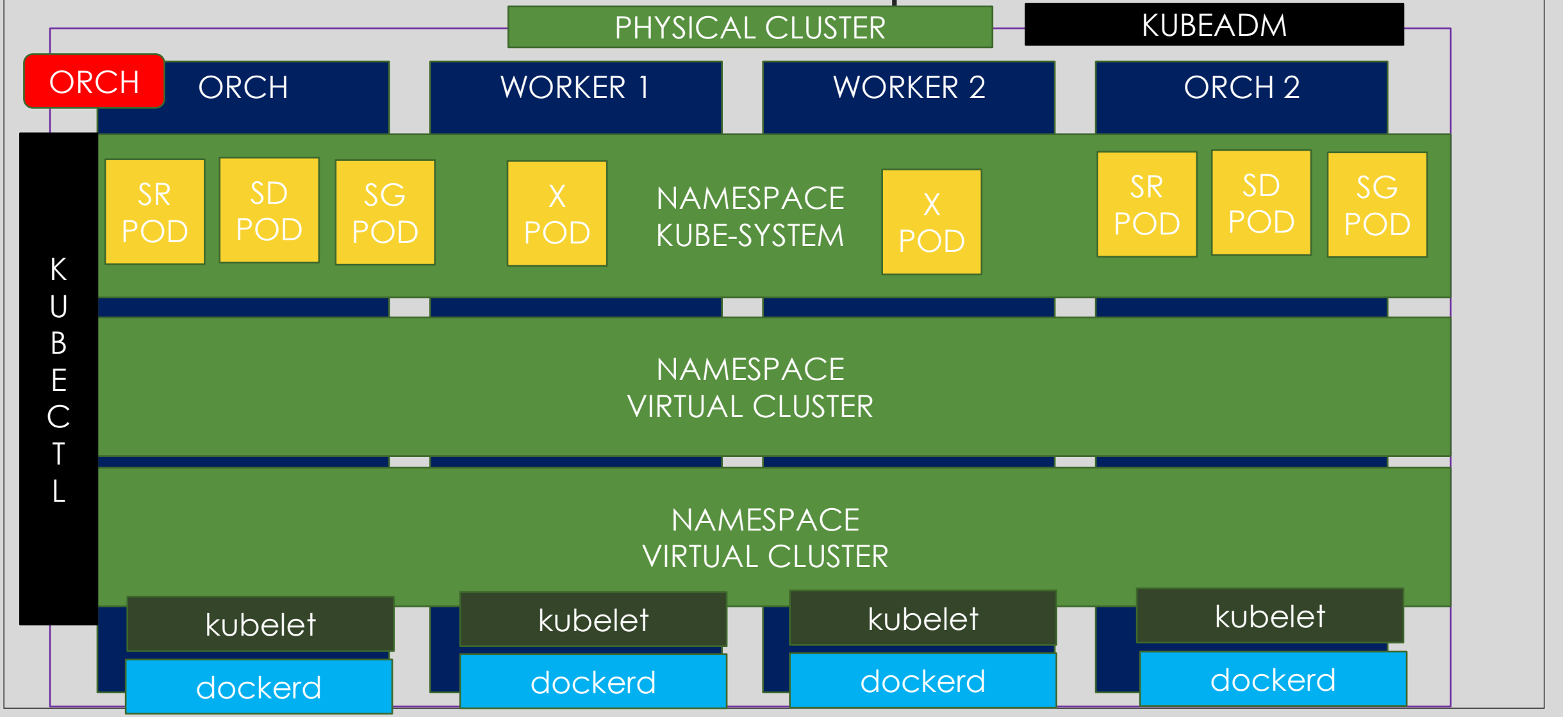
Service Registry/Service Discovery	Service Gateway	Ingress/Egress
Etcd (key value pair) API Server (service discovery)	Kube DNS (core DNS)	Gateway
Kong (oracle,mysql rbdms)	JBOSS , VertX	ISTIO , LinkerD
Zoo Keeper (no sql unstructured)	OTD (oracle traffic director)	Voyager , HA proxy, Traffeik
Consul (Cloud Foundry no sql)	NGINX Plus	Nginx , ambassador

POD

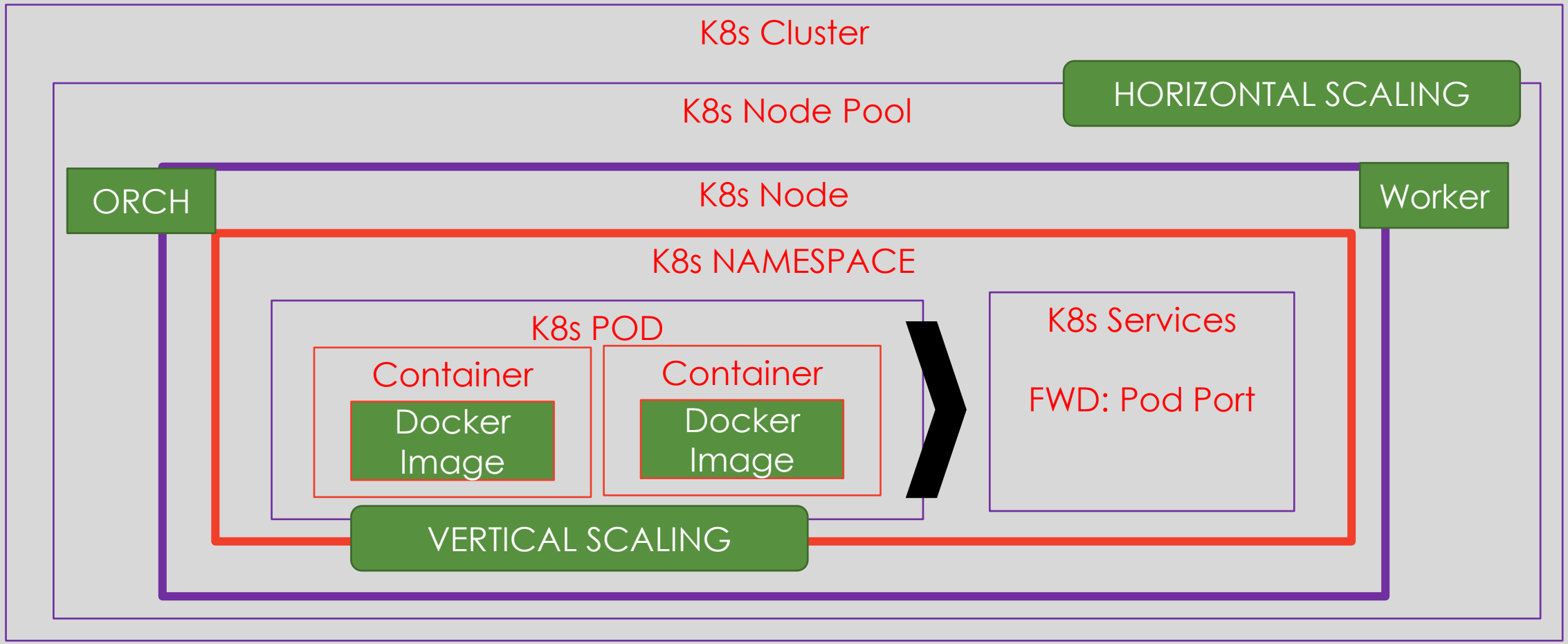


Unit of Abstraction for Scaling one or more containers exposing one or more ports (services)

Virtual cluster → Namespace



Architecture...



Properties

Docker	K8s
Containers	Pods
Containers have IP	Pods have IP
Port forwarded (static dynamic)	Services (Service Name)
Dockerd	Kubelet
Docker –machine (HOST)	Node, orch node (stand by)
Properties : JSON	Properties : Key Value Pair
Logs (container)	Logs (POD)
Restart Policy = NEVER (No)	Restart Policy = ALWAYS
Container Life cycle	Namespaced

kubectl api-resources

- Name of the object (lower case) → CLI (kubectl)
- Short Name of the object (lower case) → CLI (kubectl)
- API Group (apiversion) → YAML (import)
- Namespaced : True/False (YAML/CLI)
- Kind : Name as per YAML

Microservice Definition

- Independent by definition (Docker Image)
- Independent by Deployment (POD, # containers)
- Independent by Test (POD tested , Containers tested individually – build)
- Independent by Data Store (Different Data Sources)
- Independent by Scale (Vertical Scaling)

POD properties