
DOCKER AND KUBERNETES
dhananjayan

Breaks

- 11.15 – 11.30 AM
- 1 – 2 PM
- 3.30 – 3.45 PM

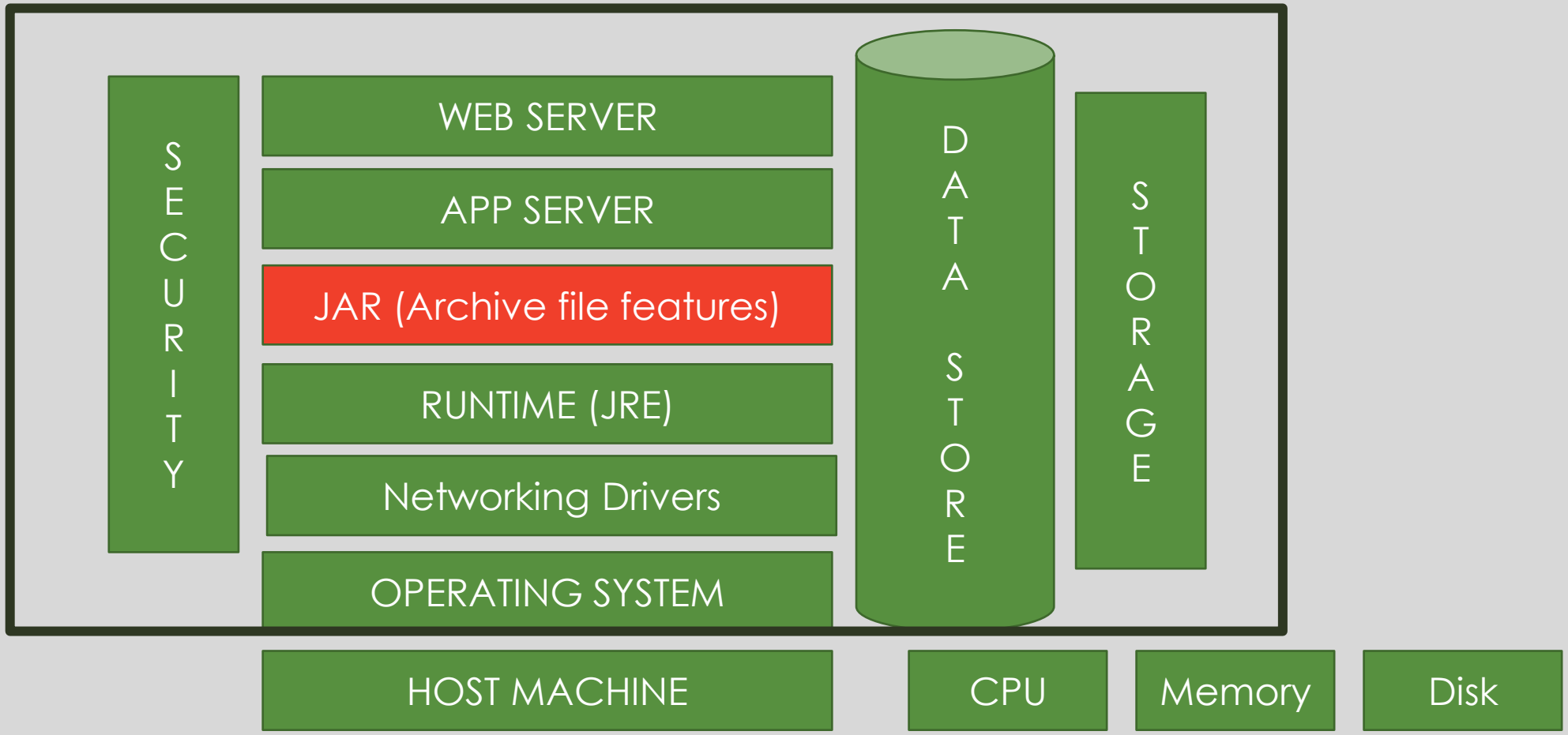
INTRO

- NAME
- OBJECTIVE
- ROLE /BU
- LINUX (ARCHITECTURE+)
- CONTAINER EXPERIENCE (+)
- CLOUD COMPUTING (AWARENESS)

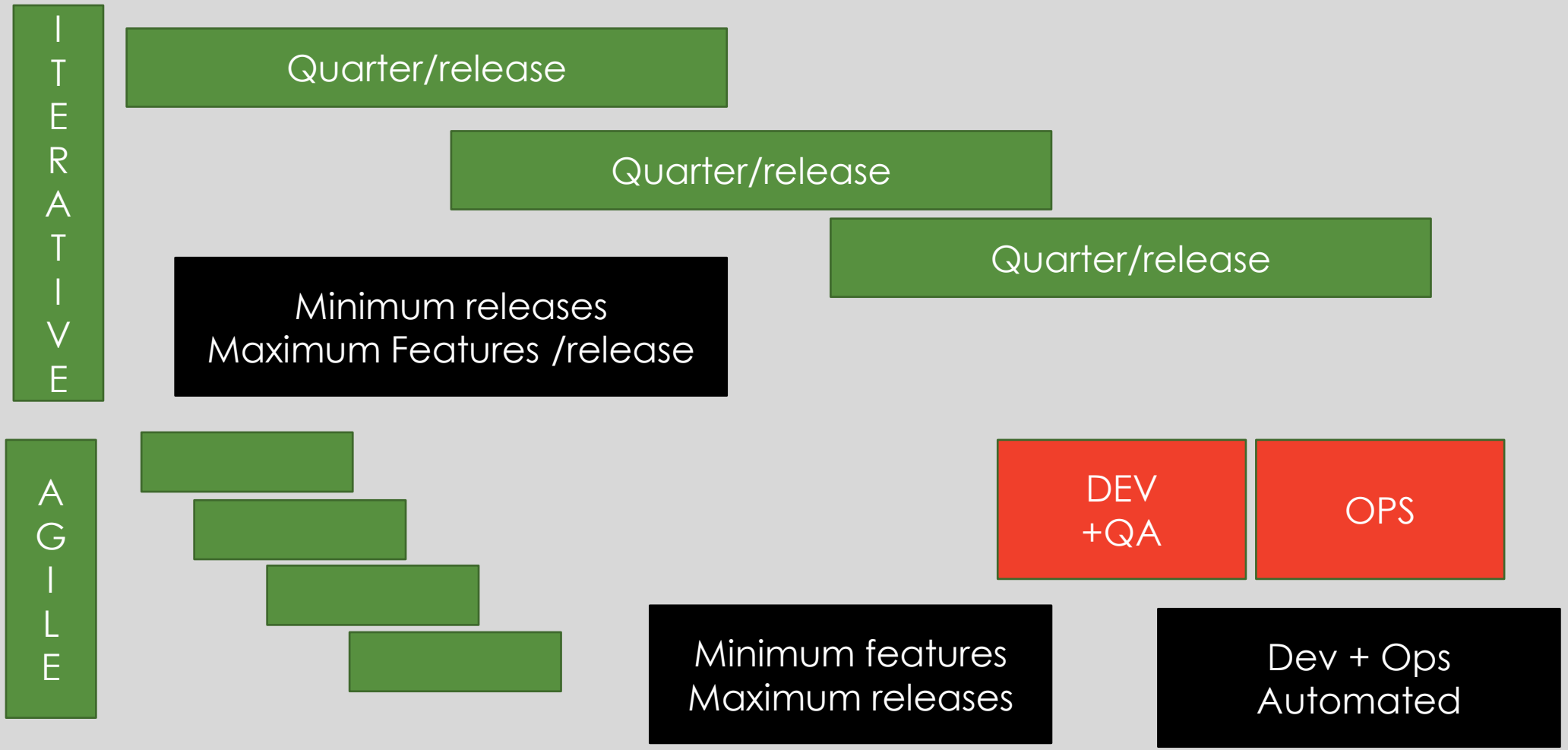
Infra pre-requisite

- No vpn
- WINDOWS/MAC → 16 gb ram
- C DRIVE – 10-20 GB HDD
- WINDOWS → VIRTUALIZATION ENABLED (TASK MANAGER – PERFORMANCE –CPU)
- VIRTUALBOX (WWW.VIRTUALBOX.ORG) → 6.0 +
- WINDOWS → UNINSTALL DOCKER FOR WINDOWS → RESTART IS REQURIED
- WINDOWS -> TURN WINDOWS FEATURES ON/OFF (HYPERV-UNCHECKED) → RESTART?
- HUB.DOCKER.COM (PERSONAL EMAIL ID)- REPOSITORY
- GITHUB.COM → CODE REPOSITORY (PERSONAL EMAIL ID)

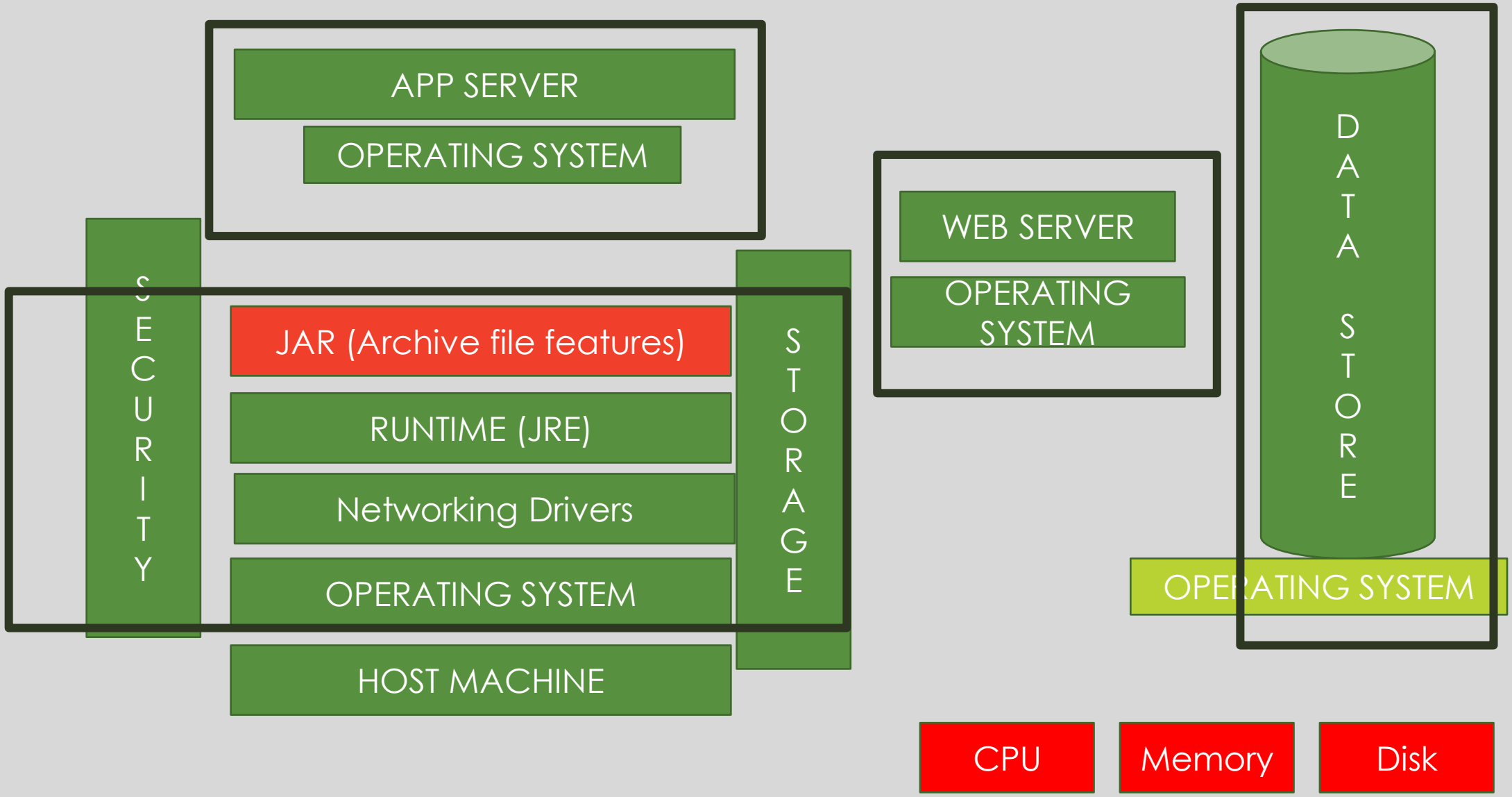
Application....



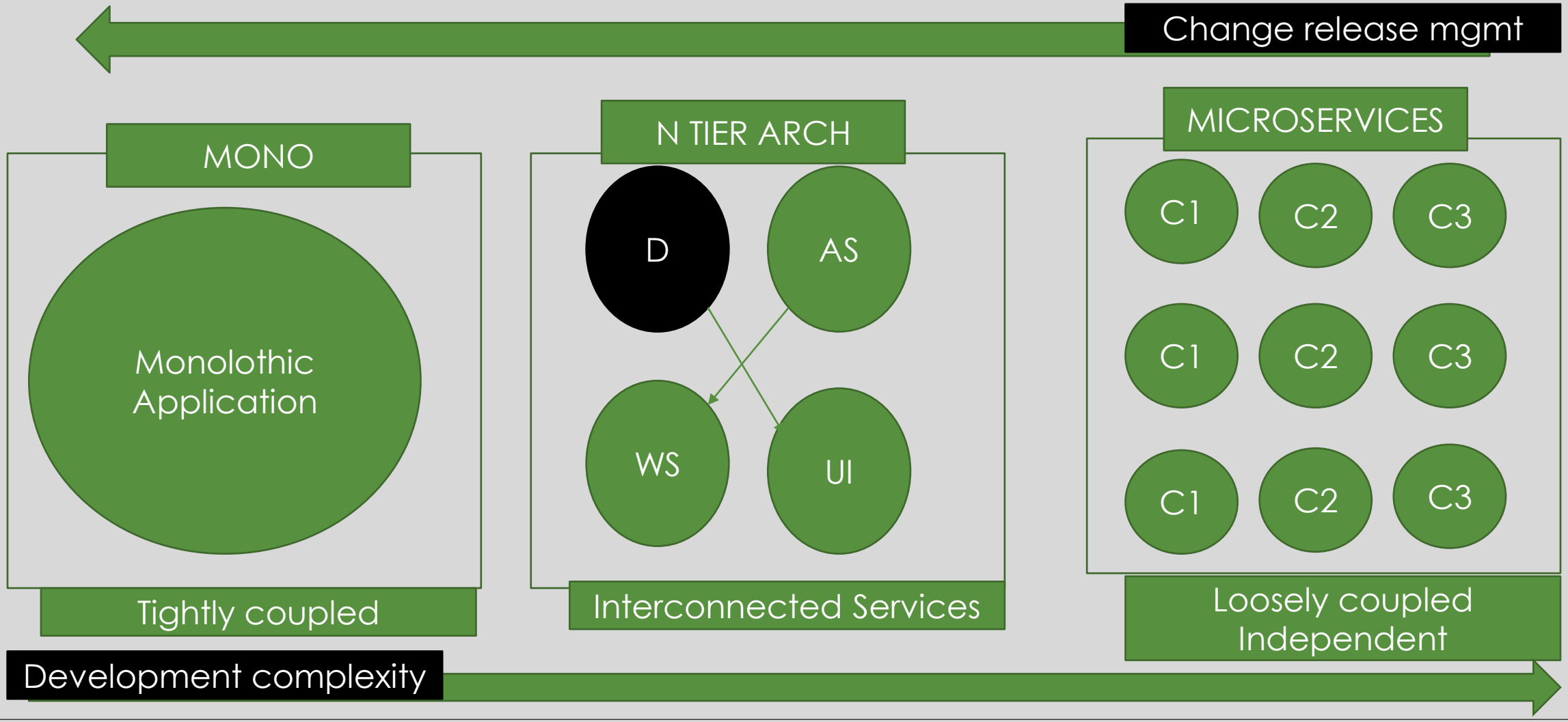
Application delivery…



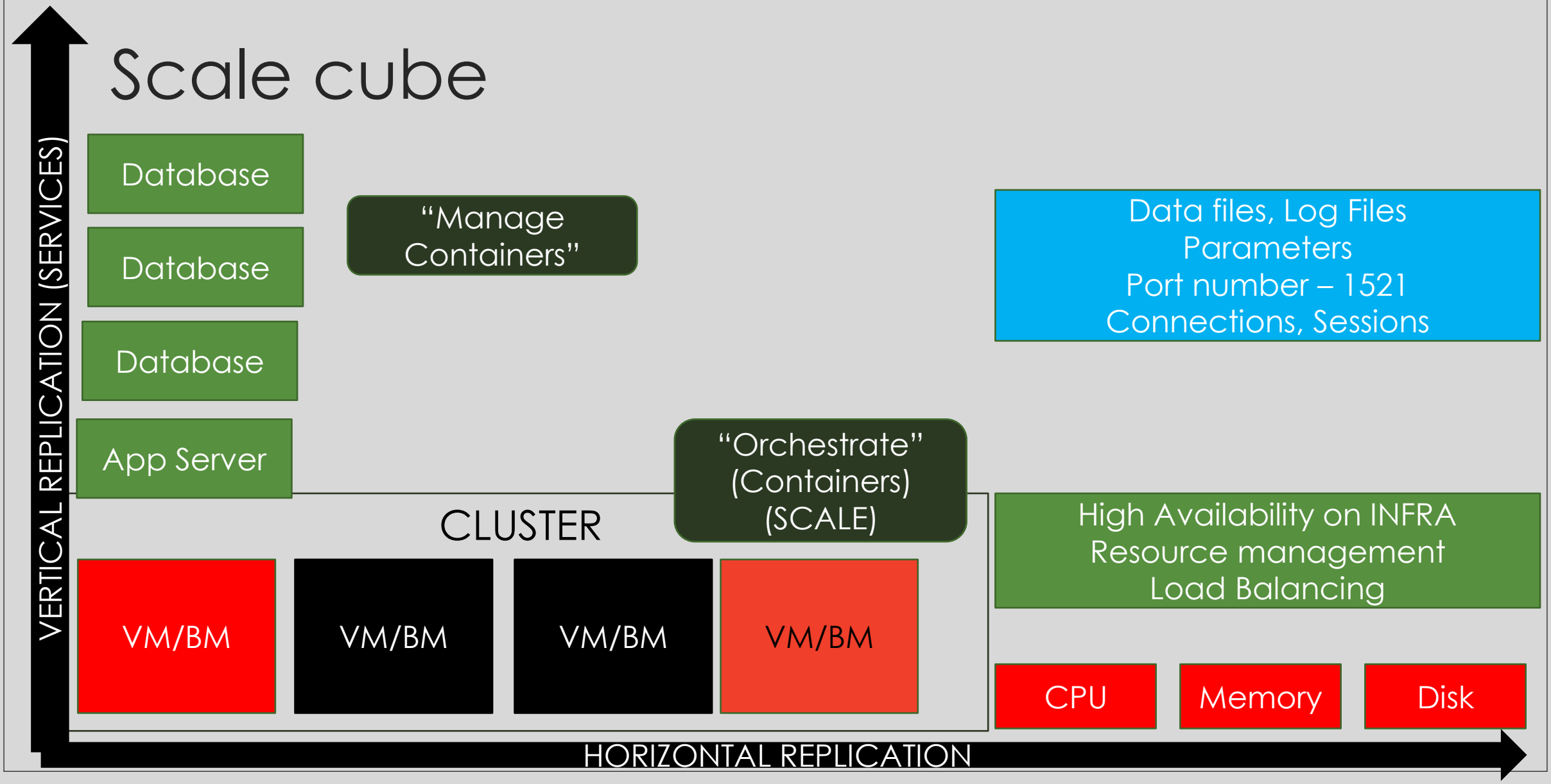
Application....



Application architecture (design)



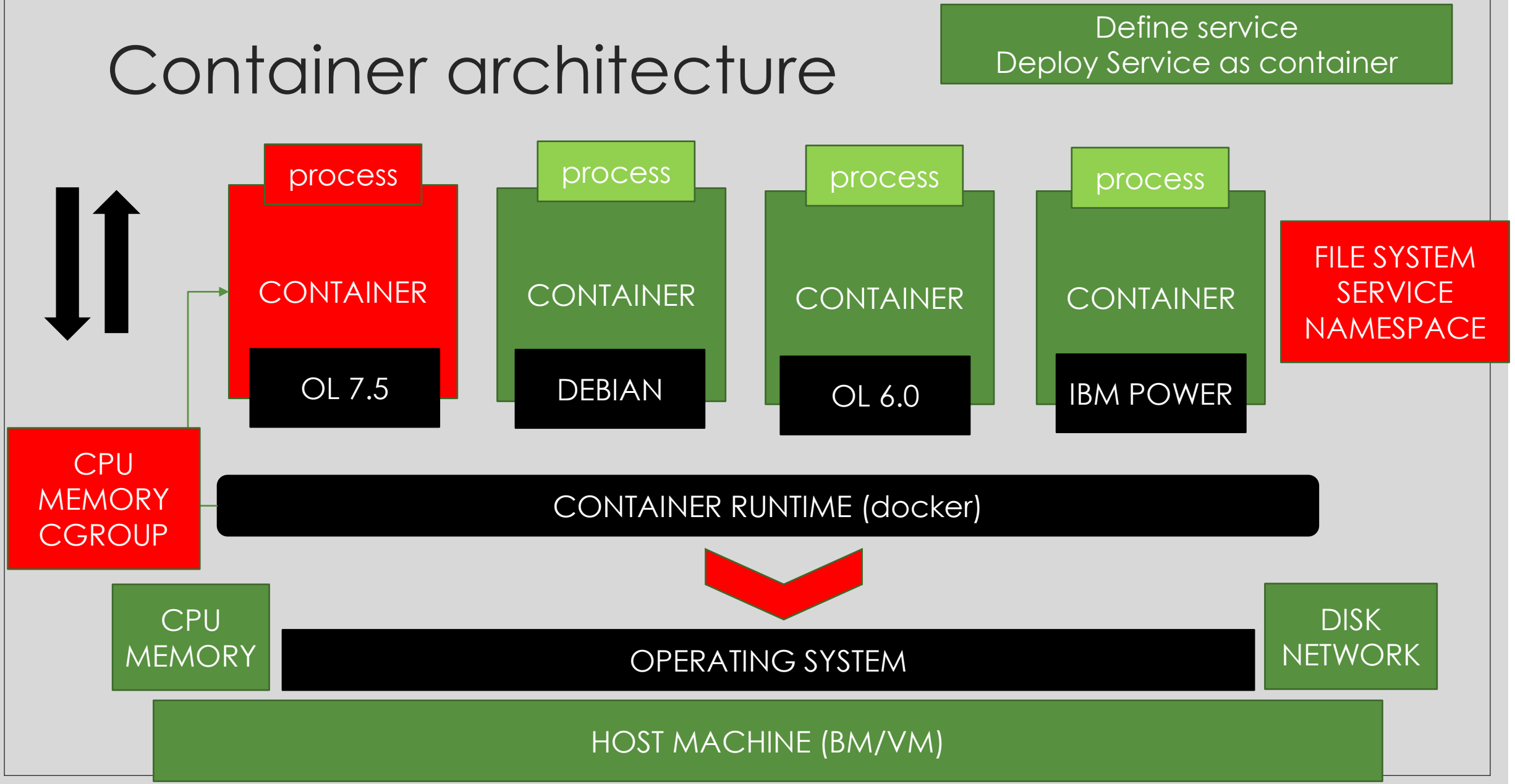
Scale cube


VERTICAL REPLICATION (SERVICES)
Database
Database
Database
App Server
"Manage Containers"
"Orchestrate" (Containers) (SCALE)
CLUSTER
VM/BM
VM/BM
VM/BM
VM/BM
Data files, Log Files
Parameters
Port number – 1521
Connections, Sessions
High Availability on INFRA
Resource management
Load Balancing
CPU
Memory
Disk
HORIZONTAL REPLICATION

Evolved…4 pillars in application

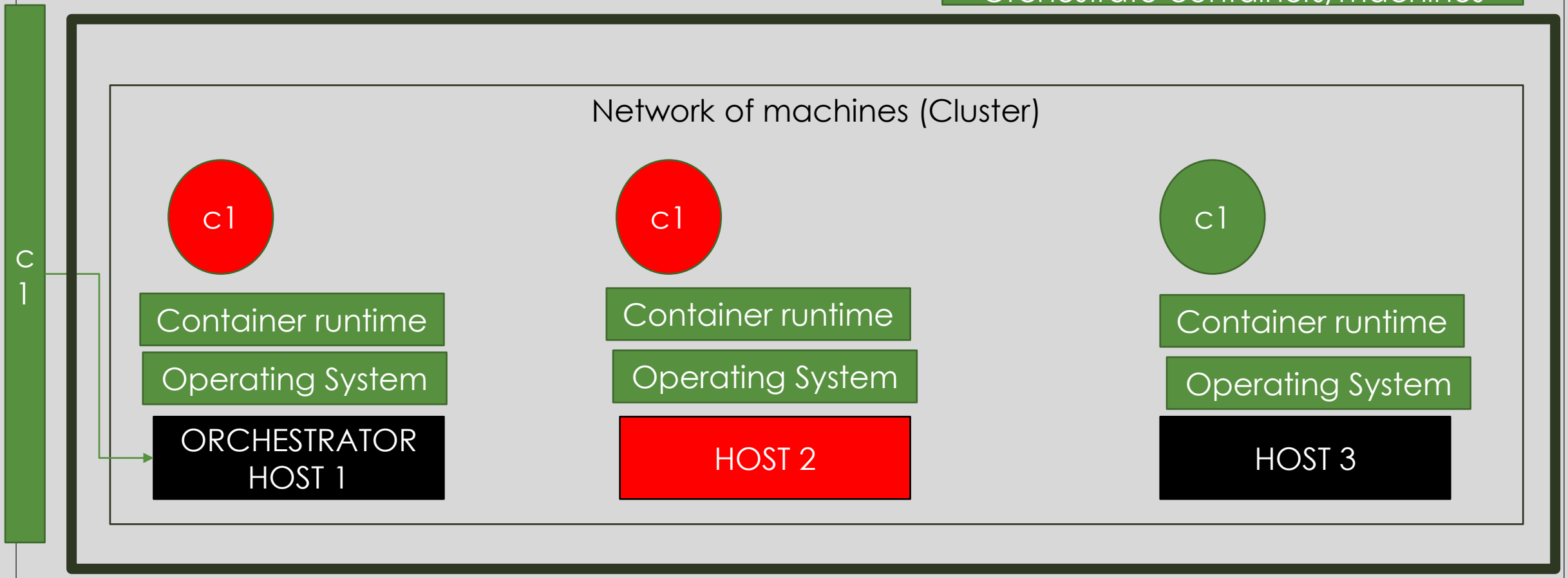
App Delivery	App design	App deploy	App hosting Integration
Iterative	Monolothic (big ball of mud)	Host machine (Bare metal)	Data Center
Agile = Dev (SCRUM)	SOA (Layer Architecture)	Virtual Machines (workloads)	On premises
Dev + Ops on Agile (Devops)	Microservices Architecture (Independent services - Deployable as containers - Scaled (vertical) - Test (containers) - Define (stack)	Container as Service Service abstraction (Separating service from underlying infra) Develop once Deploy anywhere	Cloud (OCI)

Container architecture



Cluster of container runtimes

Eradicate Single point of Failure
(SCALE SERVICE)
Orchestrate containers/machines



Docker vs K8s

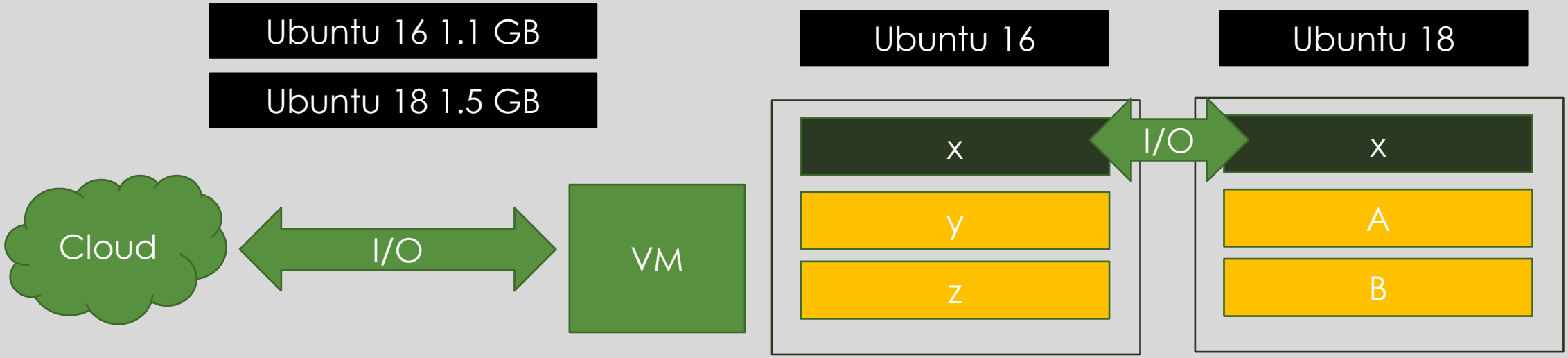
Container runtime	Cluster of container runtime(s)
Define service as blue print (portable)	Cluster of container runtimes/containers
Deploy service as containers as light weight applications (processes)	Promote High Availability (Services) – by replicating containers /machines (infra)
Mange container lifecycle	Rollout /release management (Ha)
Resource management, networking, security for containers, storage of containers, environment for the container	Orchestrate Cluster , Requests, Response, Infrastructure, Services (containers)
Maesos, Zones , LXC, OCI (open container initiative) *, Rkt, Rancher, Docker**	Docker Swarm, Marathon (Masesphere), Rancher Cloud, Kubernetes (k8s)* --Extended Docker to Cluster…

Vm vs Container

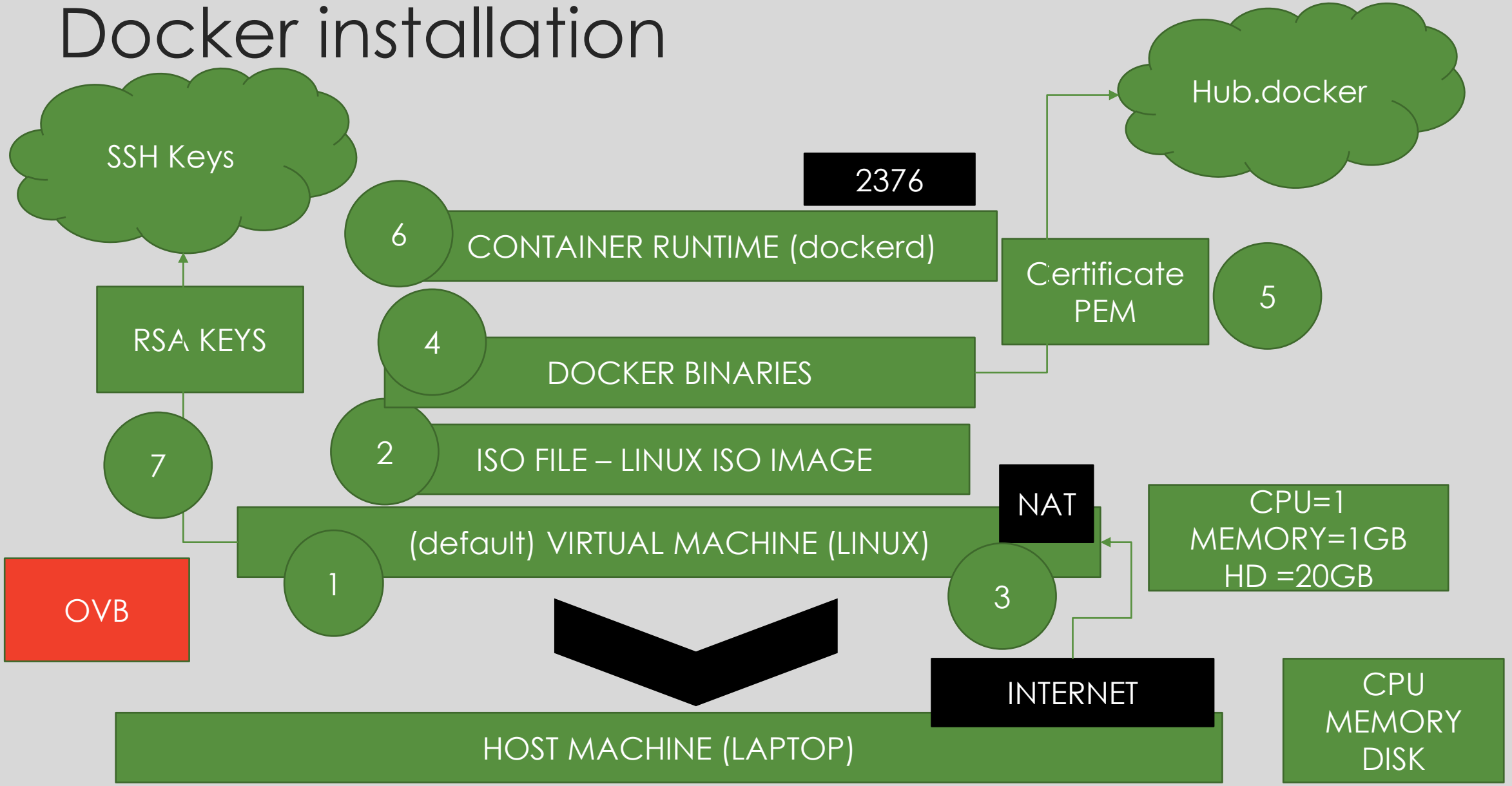
VM	Container
Blue print of VM → ISO File (Image)	Blueprint for Container → Image (portable) Image as an OBJECT (Portable)
Instance (Machine) of ISO file	Instance (process) of Image
Booting Process (init)	Listener defined in image (blueprint)
Resource management of VM → STATIC	Resource management of Container → runtime (DYNAMIC)
Horizontal Scaling (replication)	Abstract Service from Infra /Hardware Vertical Scaling (Container Services)
Monolothic, SOA	Mono/ SOA / MSA
Constant Static (memory is not released)	Running process (memory)
INIT BOOT Loading (Booting up of System)	Abstract (Lazy Loading)

Images…

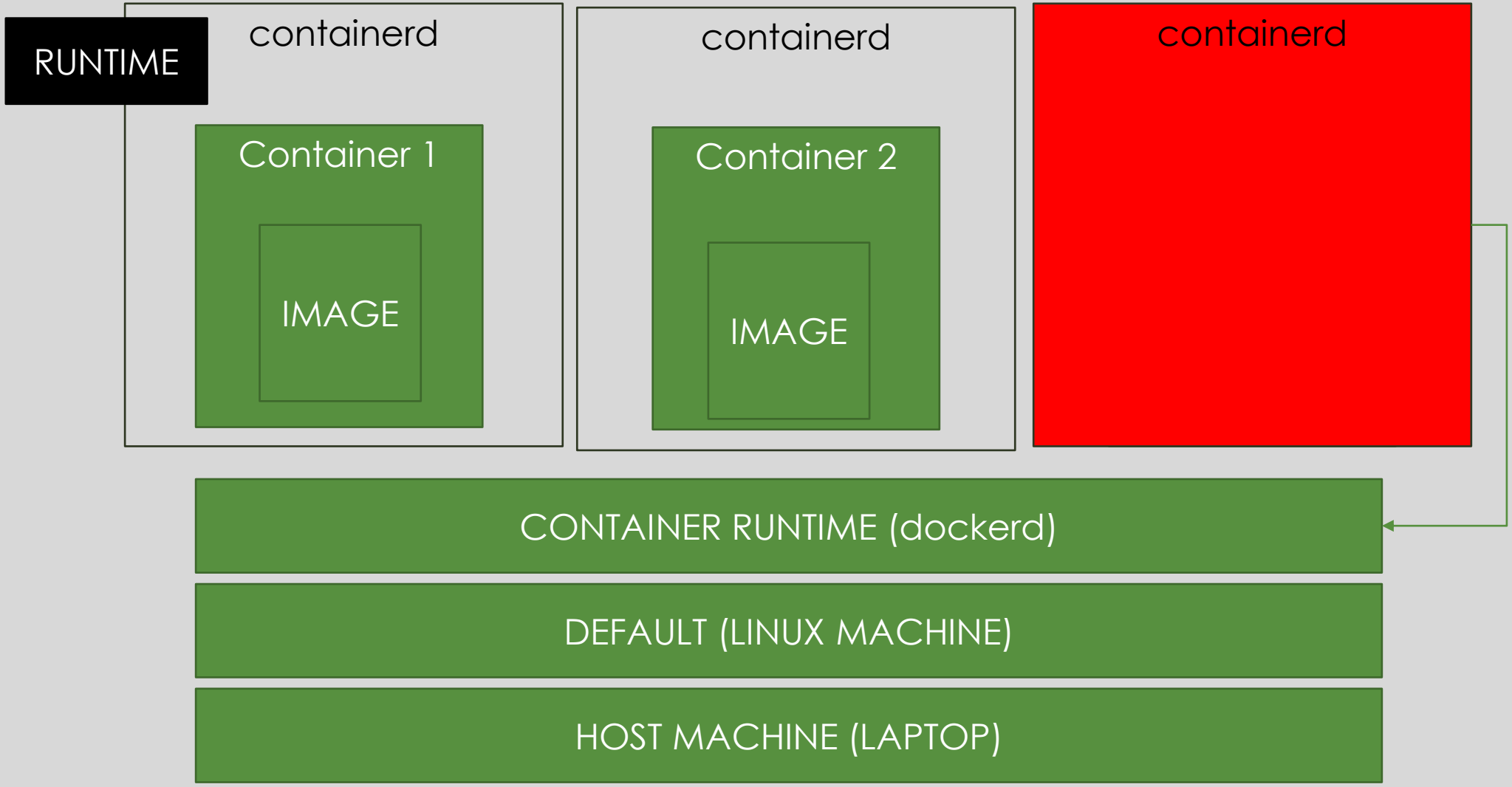
VM Image	Container Image
FILE	OBJECT (Portable)
Operating System, Dependencies, Binaries, Service(Code)and Configuration – Architecture of the Application	Light weight Kernel, Dependencies, Binaries, Code, Configuration – Architecture of the Individual Service
Full download/upload	# Layer(s), Incremental Update
File Repository (beehive)	Object Repository (hub.docker)



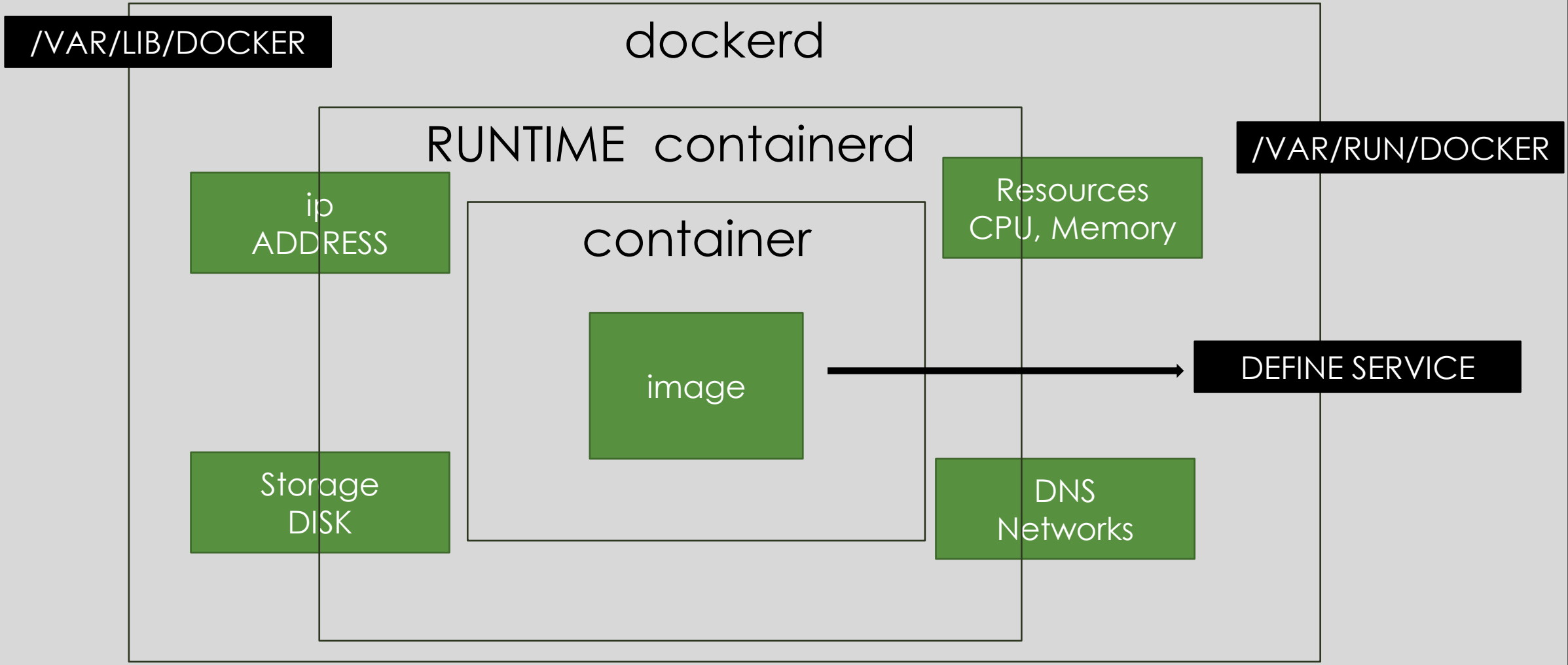
Docker installation



Mediator for container

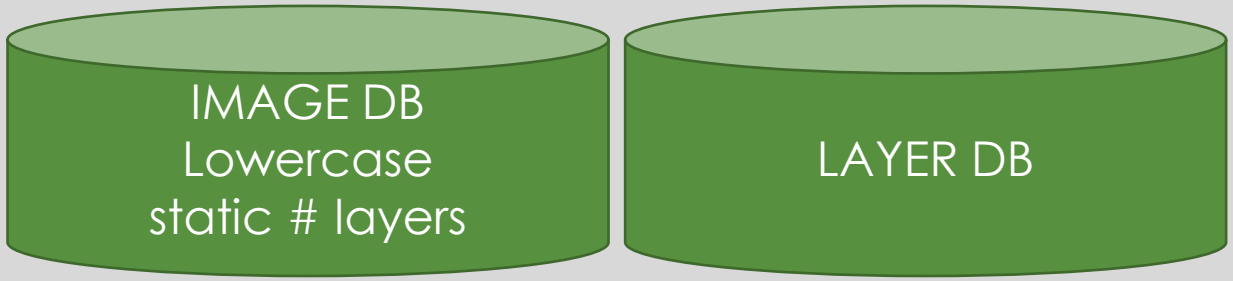


RUNTIME FOR CONTAINER - CONTAINERD



REPOSITORIES….

LOCAL	Remote	Cloud repository
Default Defacto repositories	Network Remote Image Repositories	Hub.docker.com
/var/lib/docker/image/overlay2		



Name-image

Tag-version

Repositories…

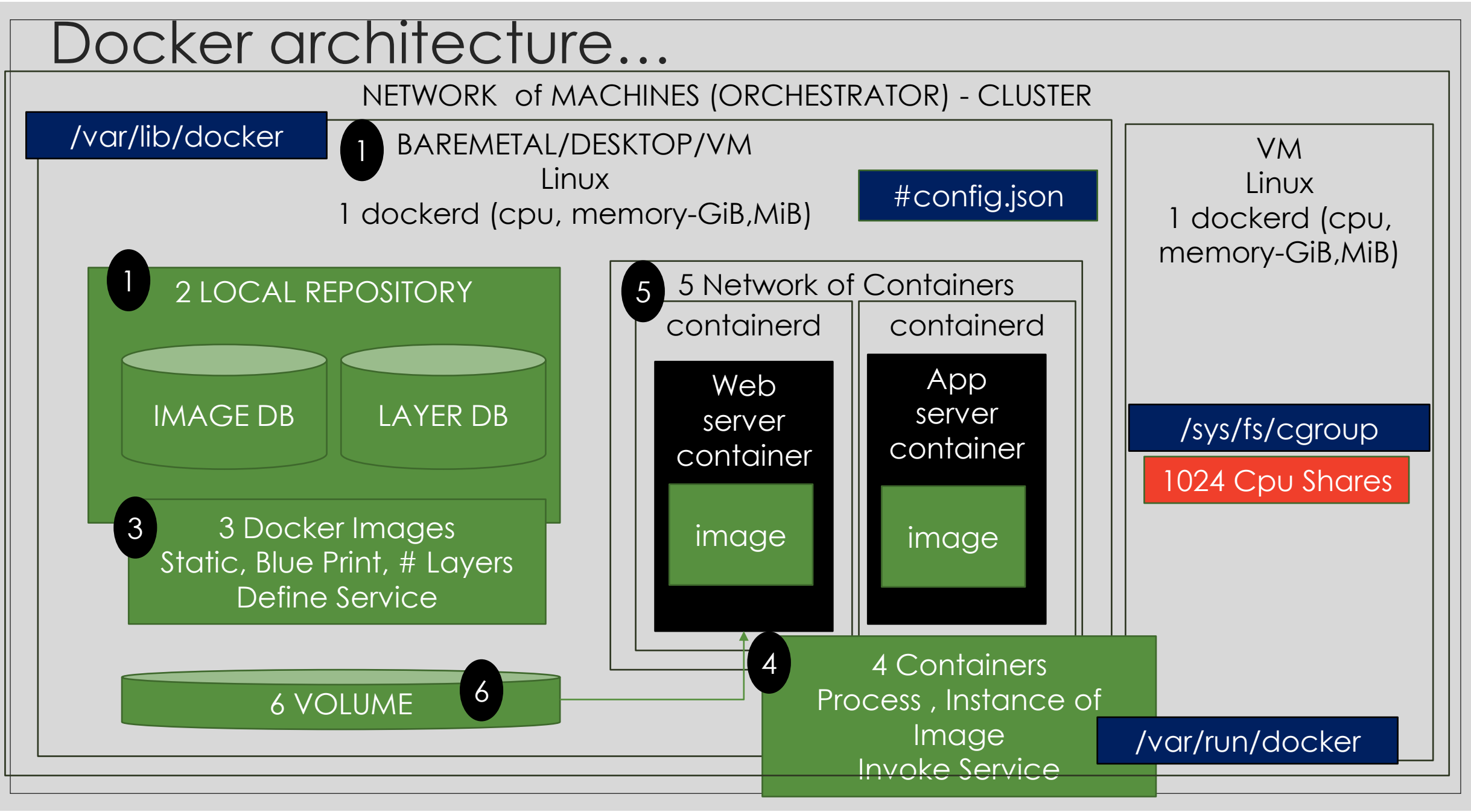
Default Docker registry	OCI V1 (ro registry)	OCIR
Hub.docker.com (index.docker.io/v1)	https://container-registry.oracle.com/	OCI Tenancy
Read write	Read only	OCI registry – read/write OCI Users

Repository-URL/name-of-image:<tag-version>

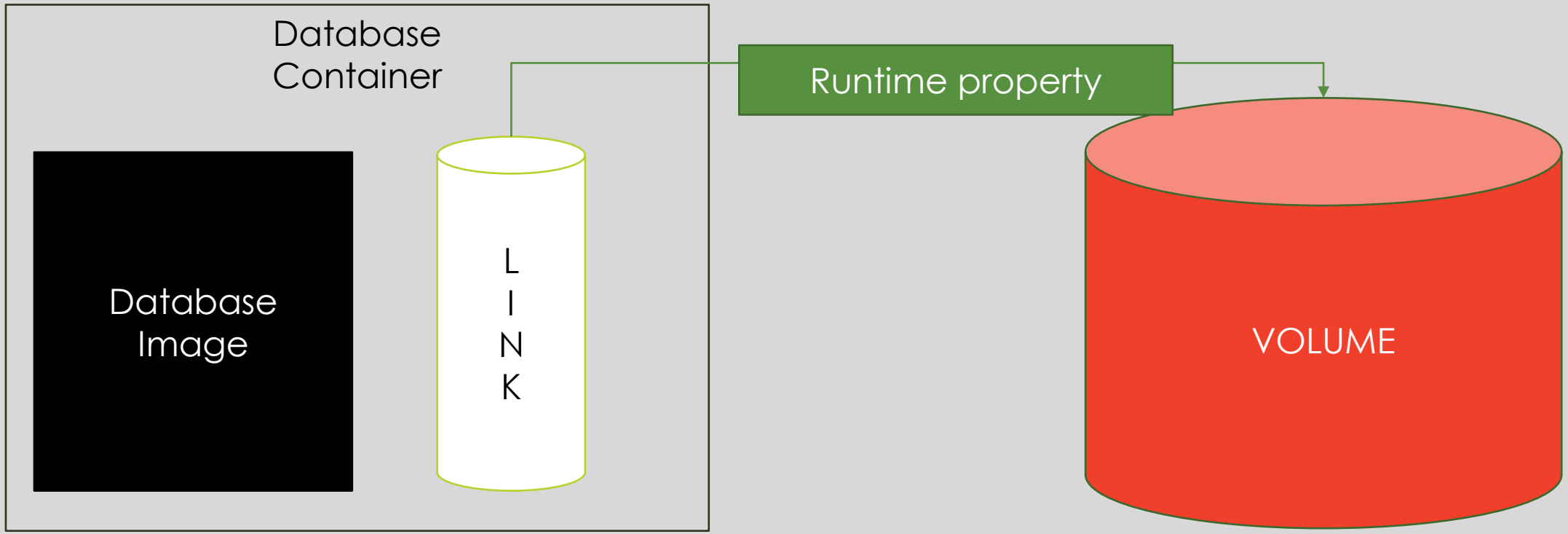
Kb/Mb/Gb (Static, Disk)

KiB/MiB/GiB/TiB proceses

Docker architecture…



Database


Database
Container
Runtime property
Database
Image
L
I
N
K
VOLUME

Docker properties read-only

Characteristics	Value
Properties of Docker Objects	JSON
Images to Secured	SHA 256 (Digest → Layers)
Automate Builds, Deployments	YAML
Storage Driver (Read/Write)	Hybrid (Objects, Files) → BTRFS ZFS,EXTFS (AUFS++)

What to look in a docker image ?

- Meta data – name, version
- Owner, Support , License
- Size of the Image (awareness)
- Architecture (OS) / Arch
- Services (Ports) , Protocols
- Parent Image (Derived property)
- Environment variables for the Image/Container
- Boot Strapper – main () CMD (Command executed when the container is started)
- Layers (dockerd)

Meta data is retained from the store repository

What to look in a docker inspect container ?

- Platform of the Container
- LogPath (/var/lib/docker/)..
- Runtime logs for container
- Log driver for container
- Port Bindings (Forwarded Ports)
- Mounts (Volume)
- Network – IP Address (172.17.0.2) → containerd
- SandBoxID : Security Context of Container

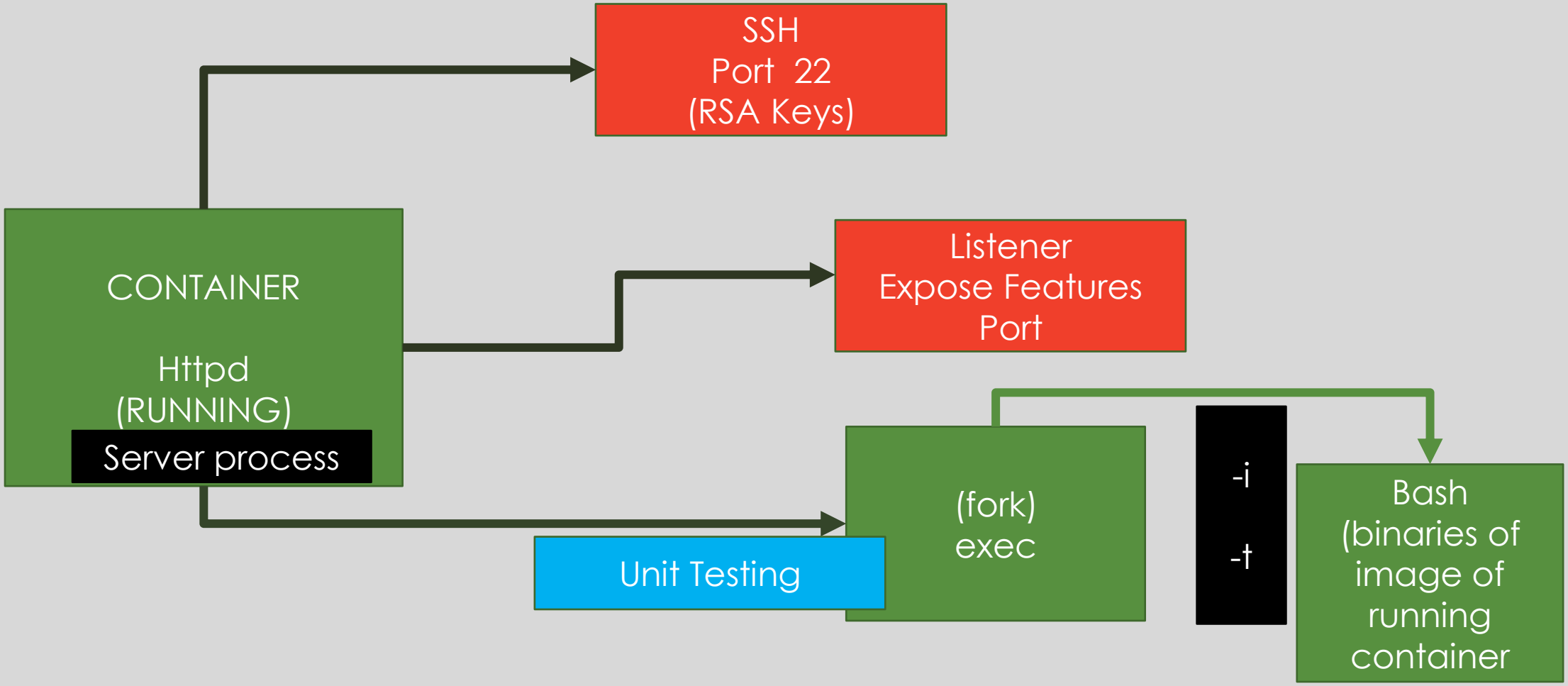
Day 2

- Access containers
- Automation with Containers /docker
- Custom Images
- Automate custom Images
- Test Images
- Use case – Application
- Use Case – Database
- Use case with Environment (Networking)
- Application scalable for changes (docker)

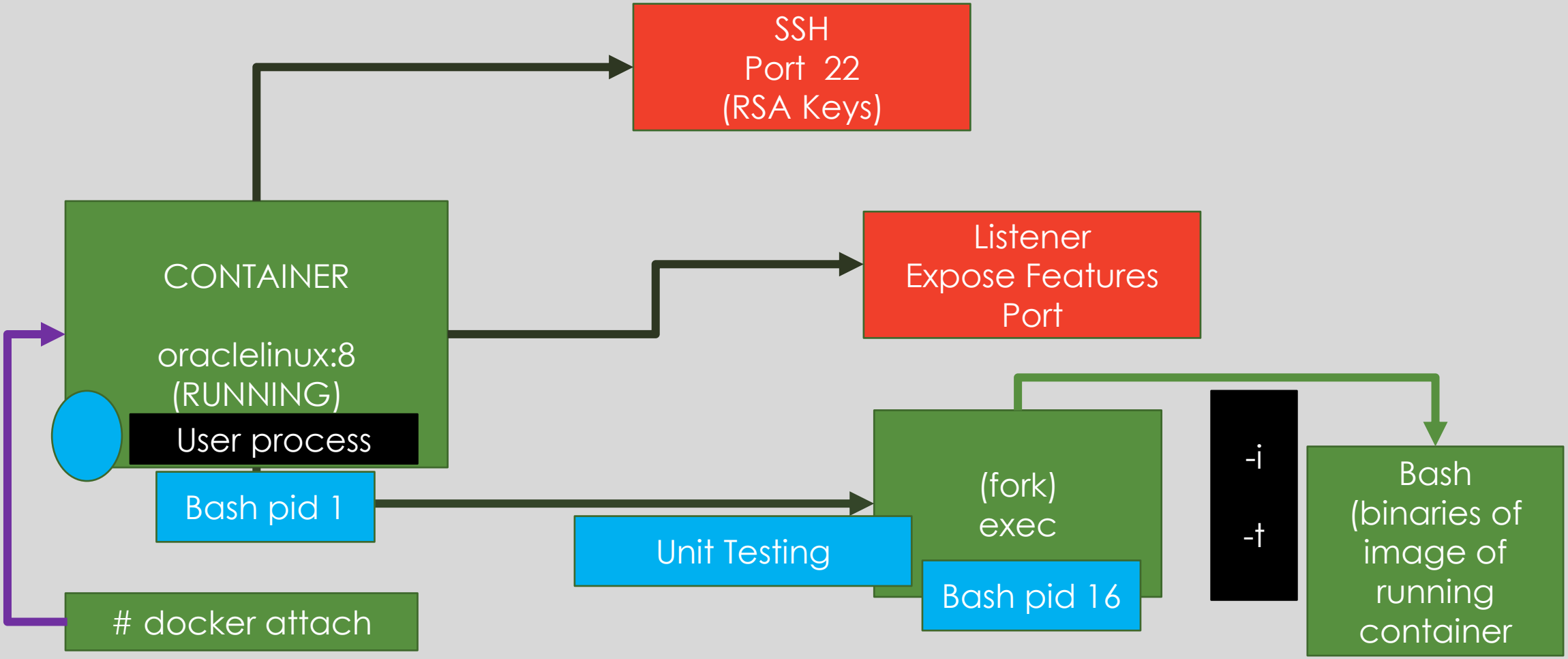
What are characteristics of application in cloud ?

- High Availability of business services (+ infrastructure)
- Rollouts (change and release management) – Ha
- Optimize Use of IT Infrastructure (resources)
- No compromise on Performance + Security

(access child) Access running Container..



(master as user) Access running Container..



CMD

“daemon”	“non-daemon”
Server Process – Listener , Instance	User Process – Shell
Communicate to API , Port Listener (UI)	Communicate to Shell – Stdin, Stdout
Httpd → httpd (server process)	Oraclelinux:8 → shell (bash)
Background process (-d) – detach	Foreground process → (-i) interactive → (-t) default stdout (tty)

Use case : when to parent, when to child

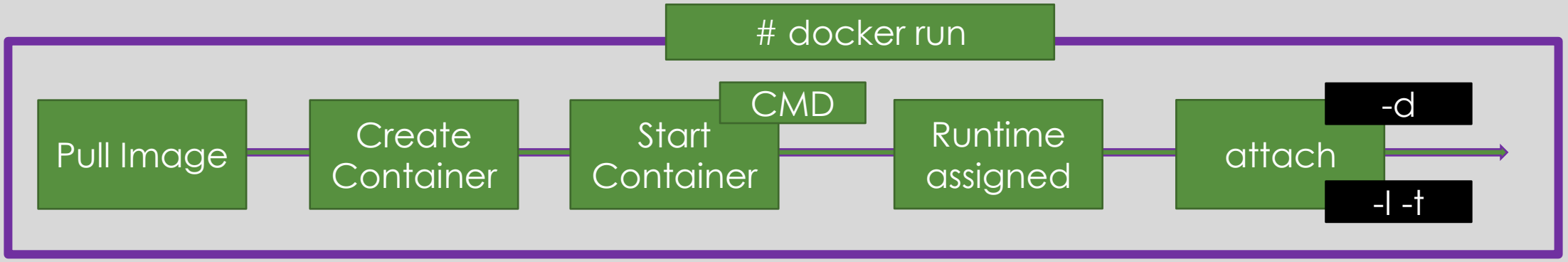
# attach	# exec
Parent process of the container (defined in CMD of the Docker image associated with the container)	Execute a fork process (child process) within the context of a container. You need to define the command to execute for the child.
Parent process → user process (attach)	Exec → execute commands, binaries and operations (unit testing) user process (-I) (-t) to exec into shell
Explicit exit → container exit. Overcome by –detach-keys	Explicit exit → only exits the child process.
Administer the container → Parent(res) Manage Process → (kill child process) Configure environments (AMC)	Logs of your App Monitor Service Status Read a File /Service (LMR)

Automations..

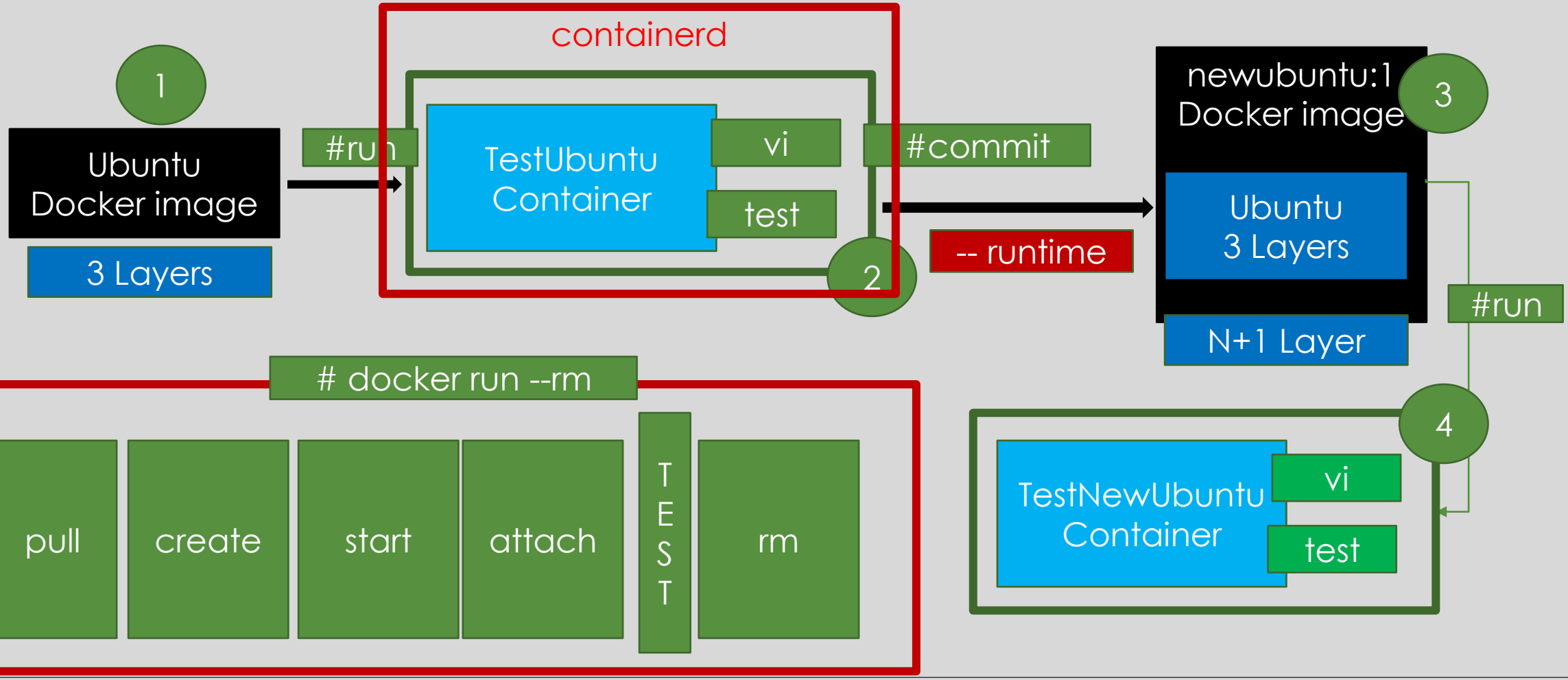
- # docker start <ID>/<Name>
- `expression` (linux)
- #echo `date` (expression)
  - Docker → $(expression)
  - # docker ps –a –q
- Extract data from JSON
  - {{extrapoliation operator}}
  - {{.PropertyName}}
  - {{.ParentProperty.ChildProperty}}

```
docker inspect -f "{{.Name}} \
{{.Platform}} \
{{.State.Status}} \
{{.NetworkSettings.IPAddress}}" $(docker ps -a -q)
```

Testing an Image



Custom image …



Build Image (functionally) ?

ARCHITECTURE
(MONO, MSA, SOA)

Packaging your Application
(Makefile or runtime dependencies)

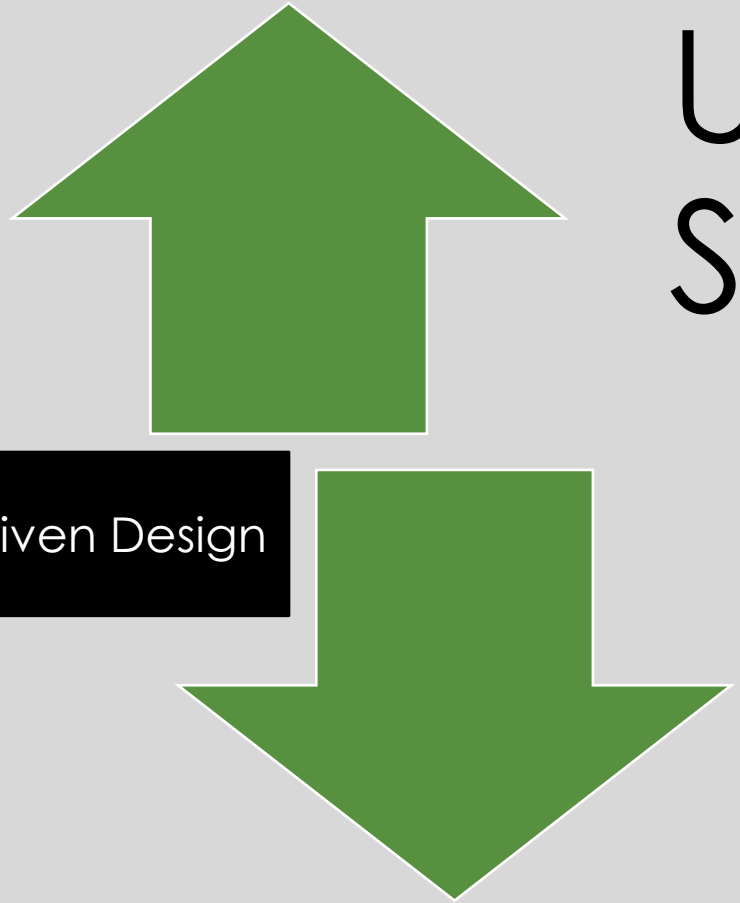
JAR

Runtime (JRE)

makefile

EMPTY
FILESYSTEM

Team structure (images)



Up Stream

ARCHITECTURE DESIGN
PACKAGING
RUNTIME VERSIONS

Domain driven Design

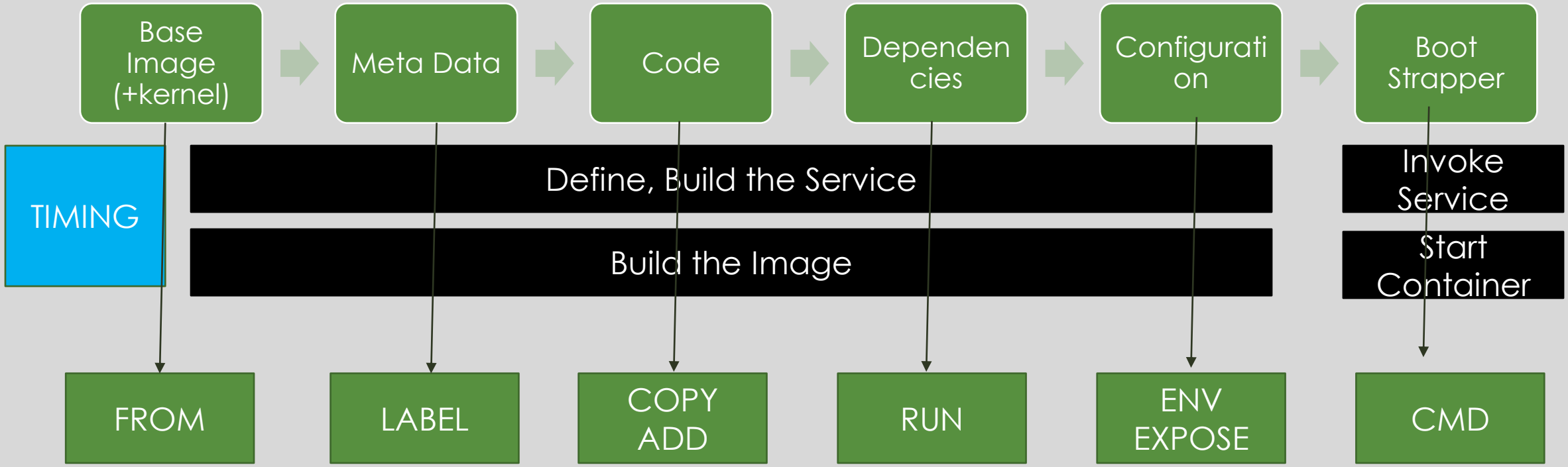
Down Stream

IMPLEMENT
API
UNIT TESTING

Custom Images

# docker commit	Dockerfile	# Rancher
Converting Container FS to Static Docker Image	# docker commit	Not Standardized
Layers : N+1	Choice of Layers → Publisher	Not Free
API : No (No defined API)	Well defined API for Image builds	Root File System of VM is converted to Docker Image
Foreground Process	Background process	Not all kernels are supported
No Support of Automation	Support Automation Maven, Gradle, Jenkins , Wercker	VM Root FS to Docker Image
Manual deletion of Temporary Objects	Automatically Prune (delete) temporary objects	

Dockerfile (Docker Image)



Use case: Dockerfile (Install Vi)

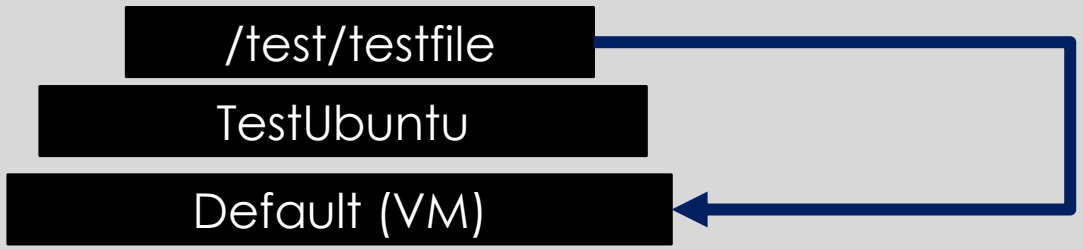
Testfile
(TestUbuntu)
docker cp

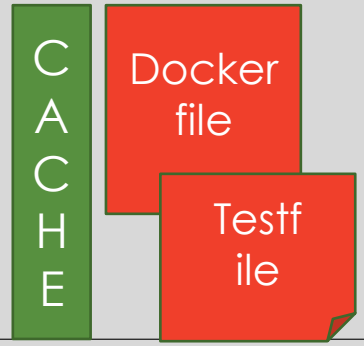
Dockerfile

Docker image
build

Docker Container
(Test)

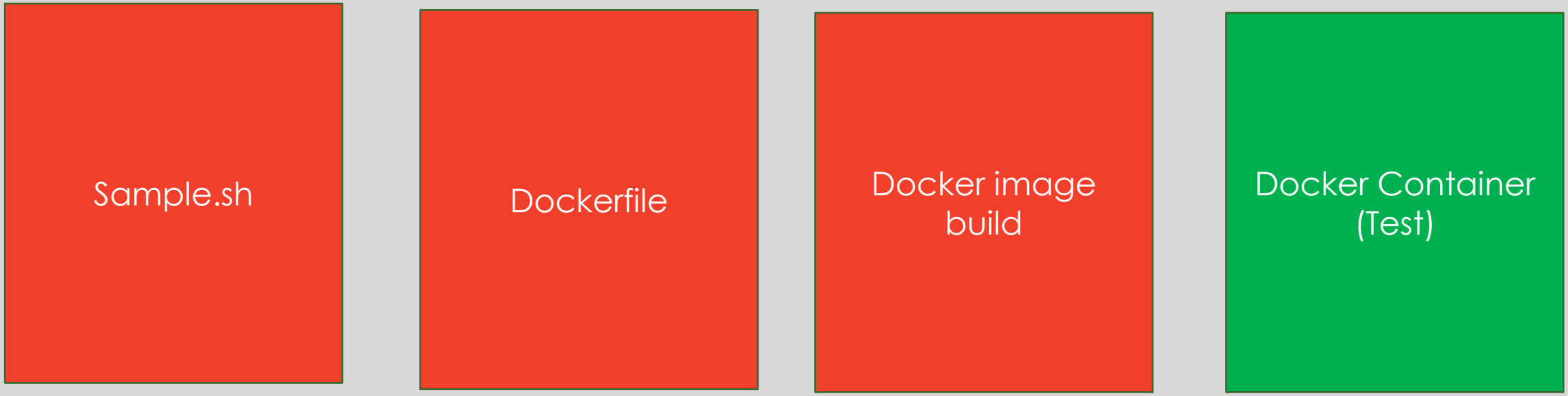
docker build –t <image-name-lower-case> -f <Dockerfile-name> [file-cache]



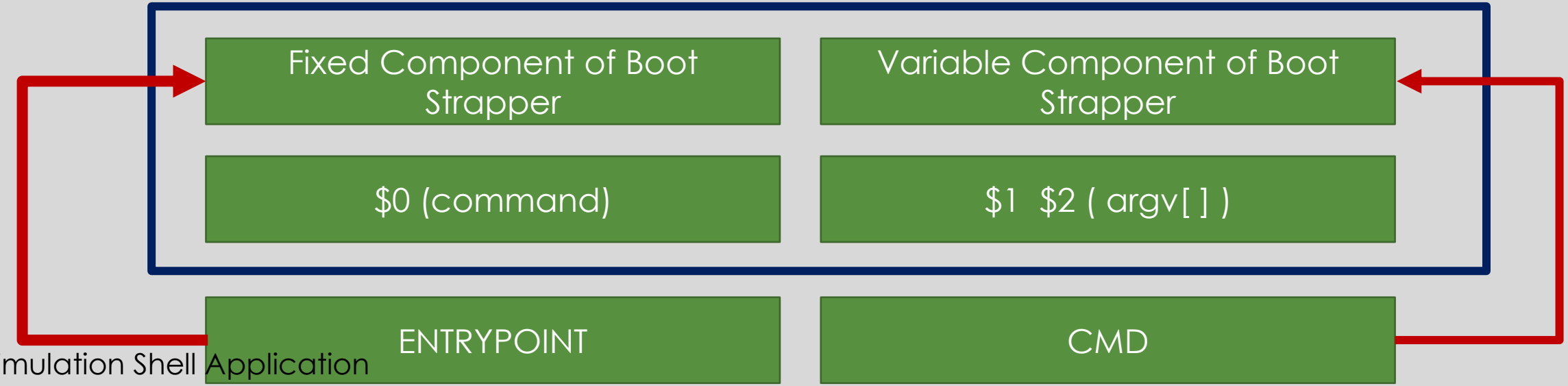


Use case : 2 (Shell Application)

1 Undermine the Image (no compromise on usage of image)
2 Abstract the Source of the source code



Mitigating Use case risk (slide 40)



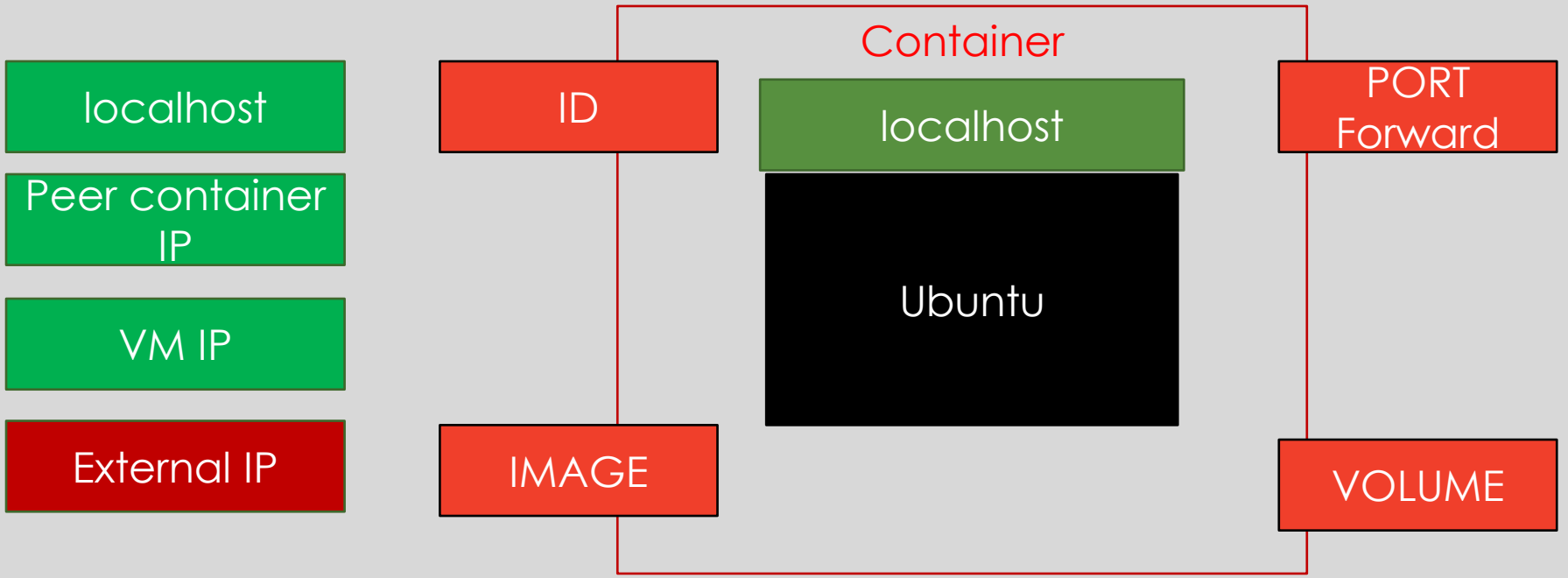

```
# Simulation Shell Application
FROM ubuntu
LABEL MAINTAINER appservices-shell@oracle.com
COPY Sample.sh /code/Sample.sh
COPY testfile /code/testfile
RUN chmod +x /code/Sample.sh
ENV version=2
ENTRYPOINT ["sh","/code/Sample.sh"]
CMD ["/code/testfile"]
```

Docker image

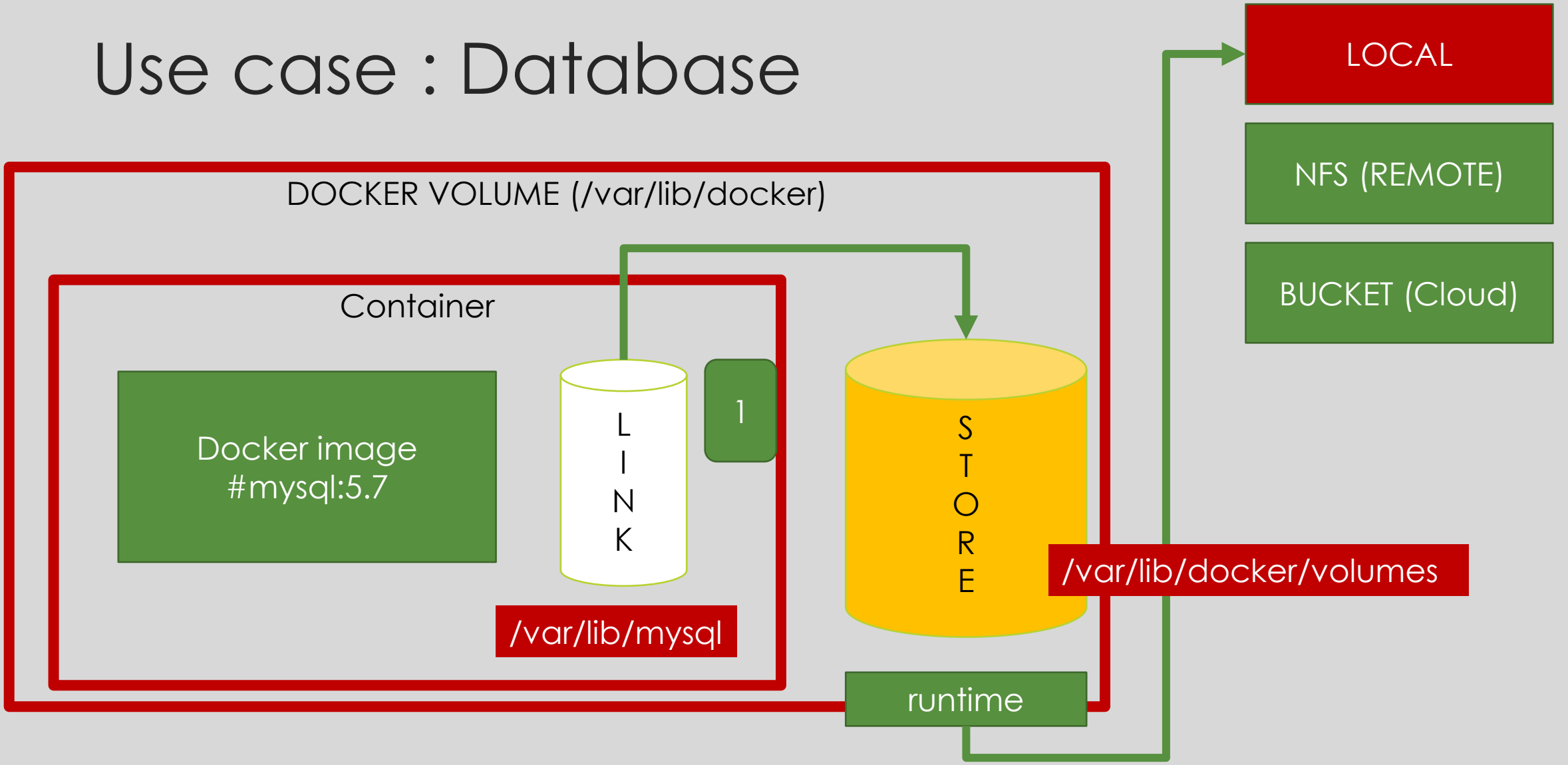
Docker Container

Use case : EntryPoint (Networking)

- #ping –c3 **localhost** (container-ip)
- #image : ubuntu:trusty



Use case : Database



Database Implementation

Image Build

Container Create

Container Start

3 ENV MYSQL_ROOT_PASSWORD (CONTAINER)
ARG (IMAGE)

2 #createdb.sql

3 Database Starts Initdb.d

Mount

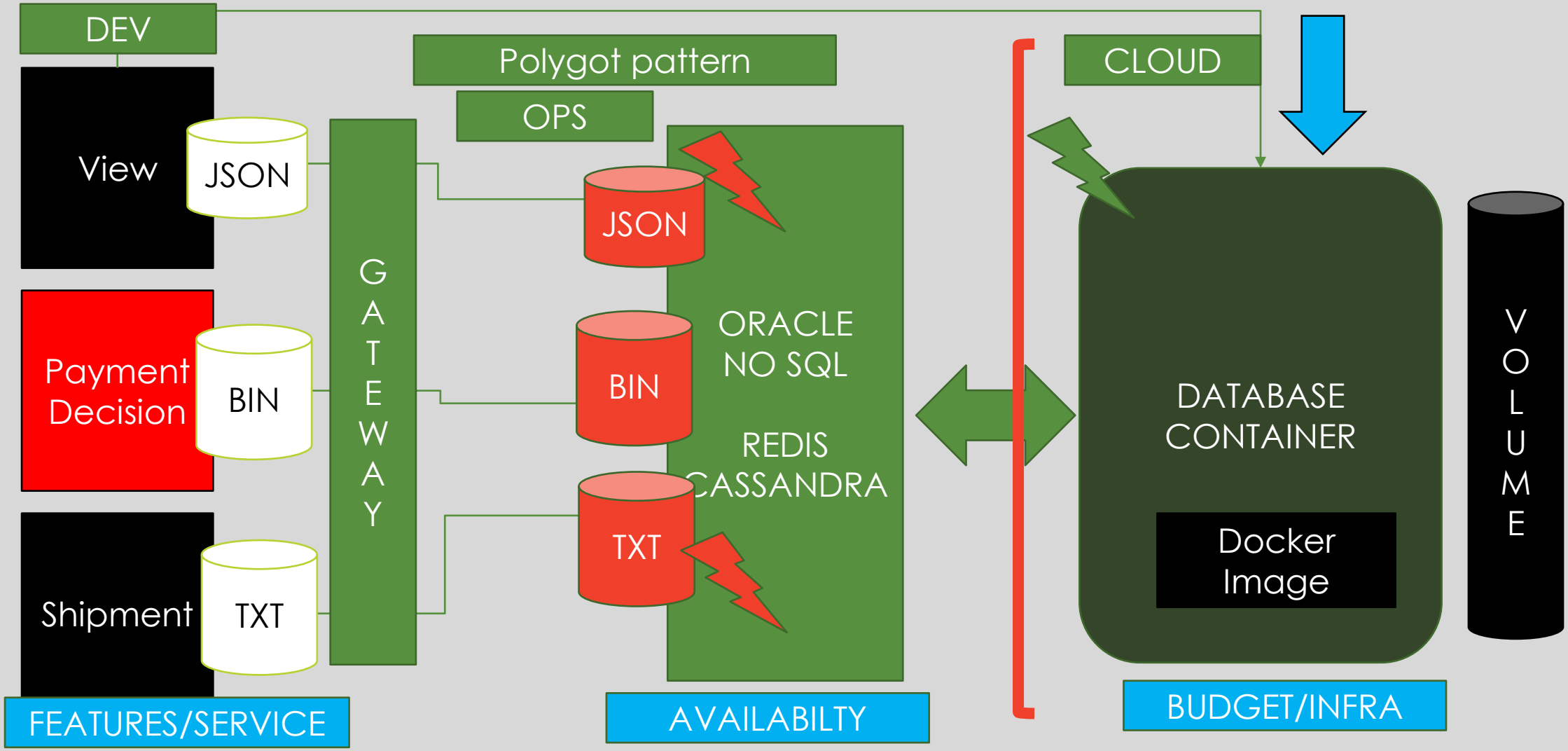
Ready Conn

/docker-entrypoint-initdb.d

Day 3

- Database – MDB
- Troubleshooting Containers
- Sharing Docker Images
- Service Management Containers (Ports)
- Communication Patterns
- Volume Management (Simple)
- Networking with Containers
- Architecture Use case
- UI Containers (OJET)
- Container Policies.

Microdatabases... (applications)



Troubleshooting…

EDA

Container specific Logs

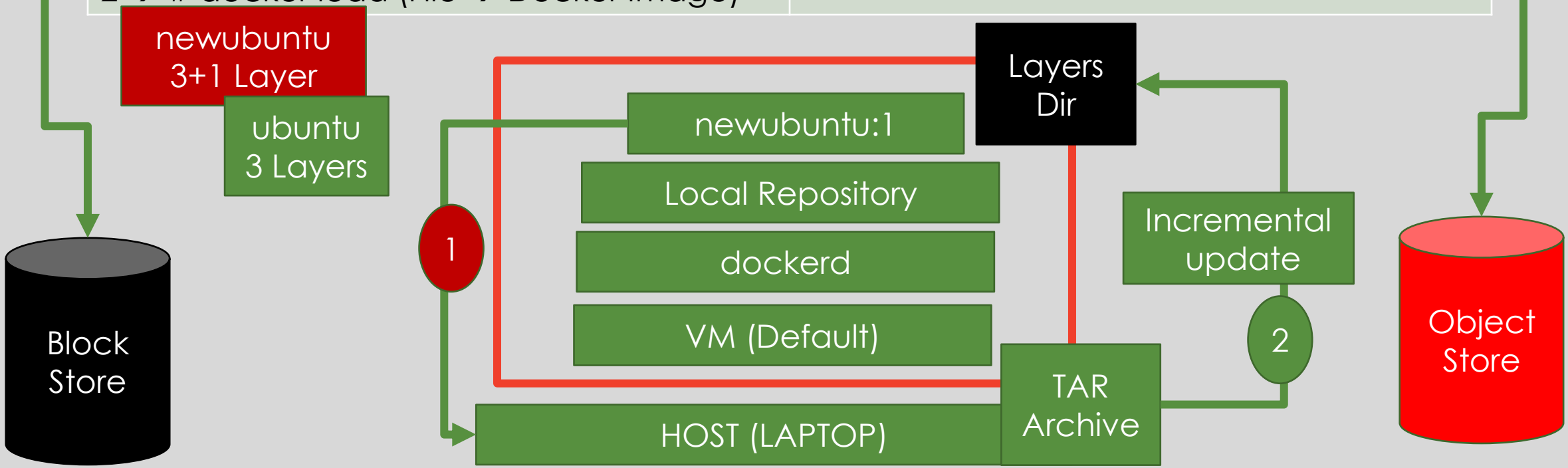
Process Logs	Admin Logs	File Logs	Event Logs
Running in Container What ?	Scrutiny Who/When ?	Audit What has changed?	Admin of Docker (around docker)
Parent process of the container (Core CMD/EP)	Parent process (Service) of the container	File System Changes A , C. D	Events (docker events)
# docker logs	# docker inspect LogPath…	Parent, child process changes to F/S	#docker events –since=30m

CFS

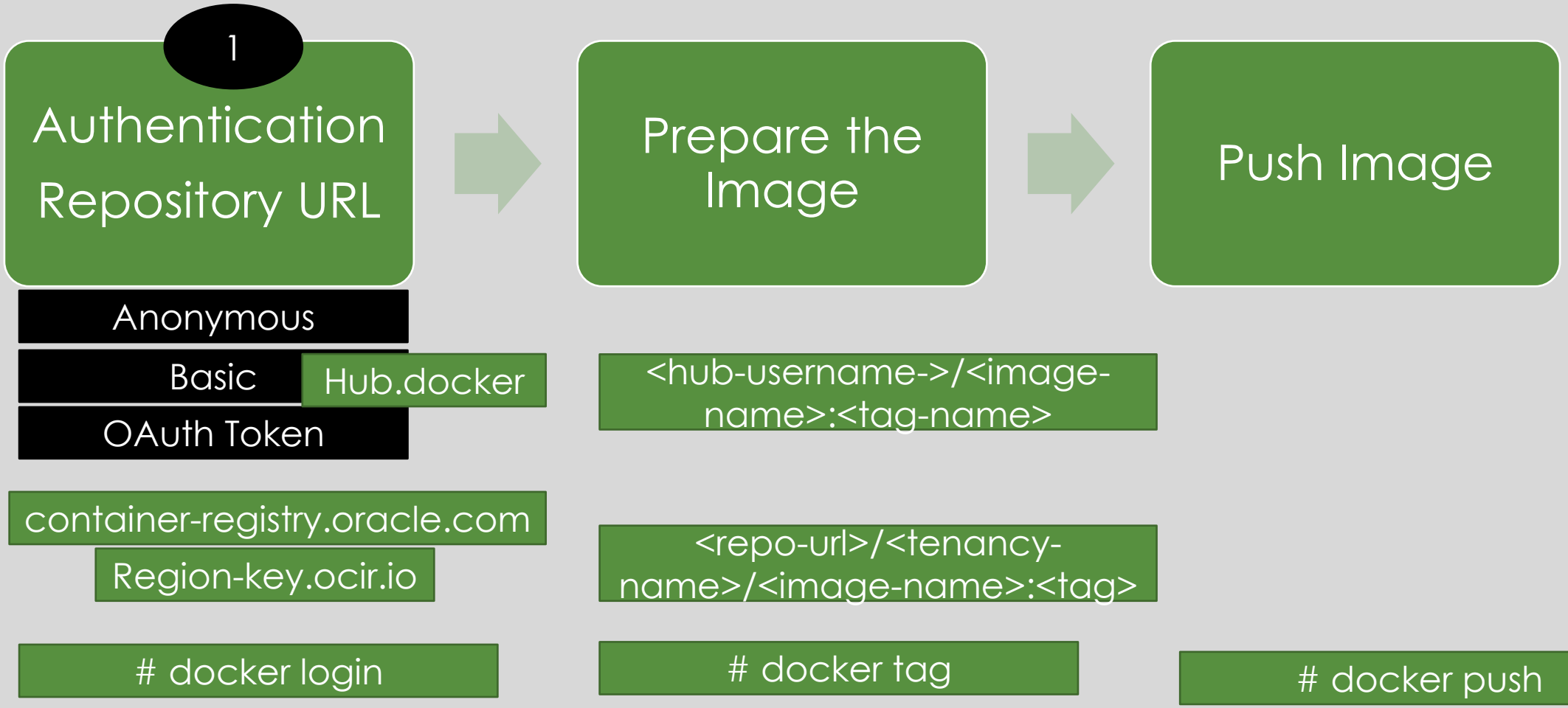
IFS

Share Docker Images

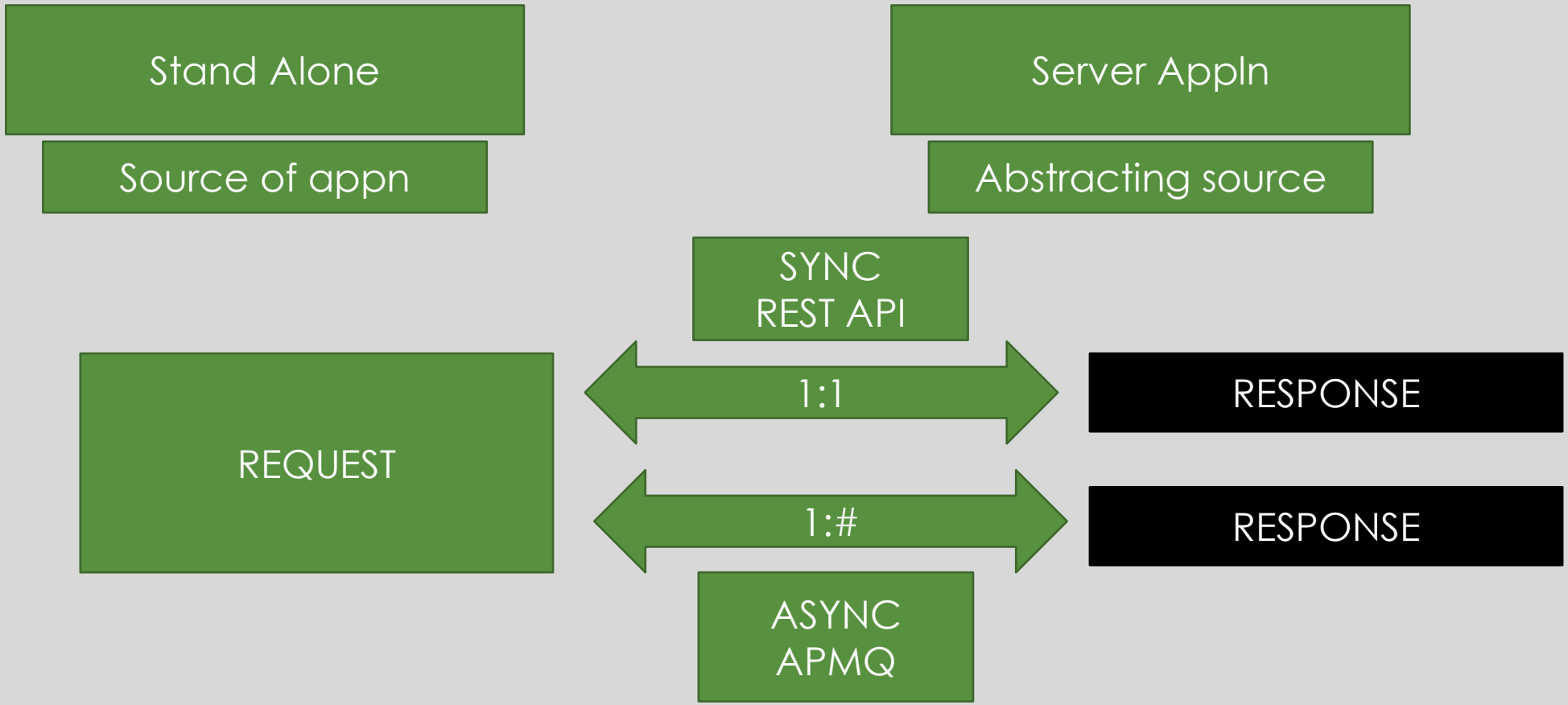
Via Data Center	In Cloud Repository
Serialization : Object → File Meta Data for an Object is retained 1→ # docker save (Docker Image → File) 2 → # docker load (File → Docker Image)	Save the Images as Objects. Refer Slide 49



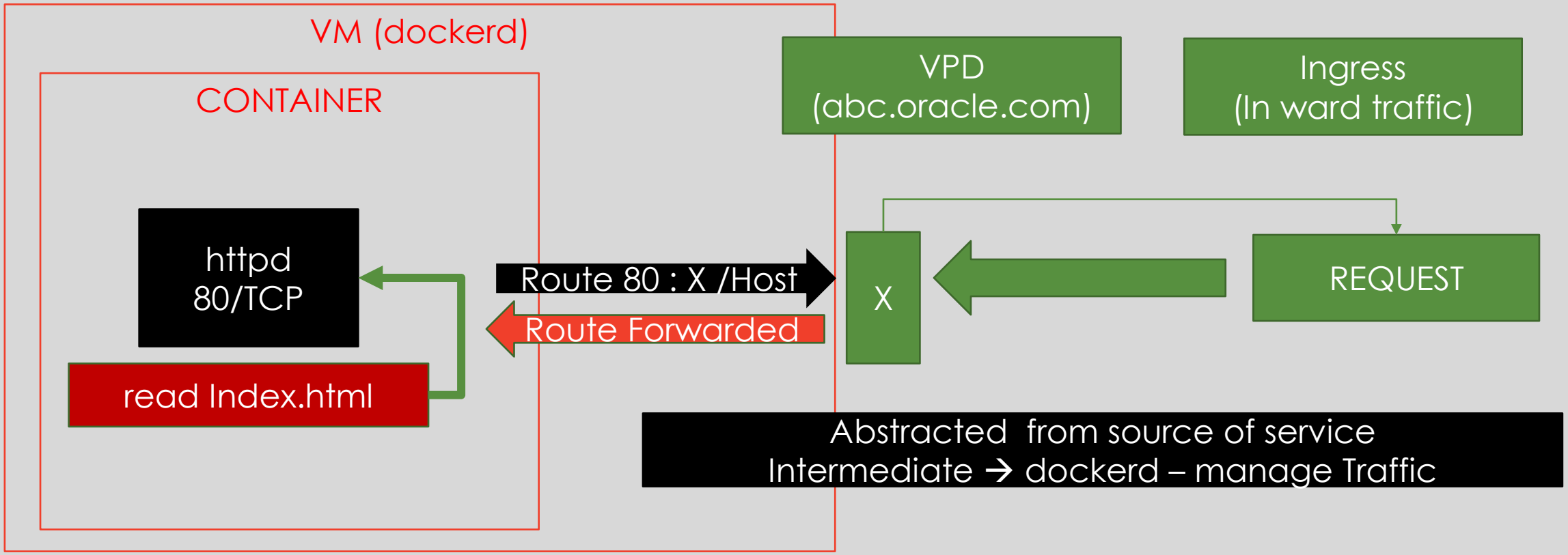
Share Images in Cloud



Service Management



Port forwarding Service (REST Sync)

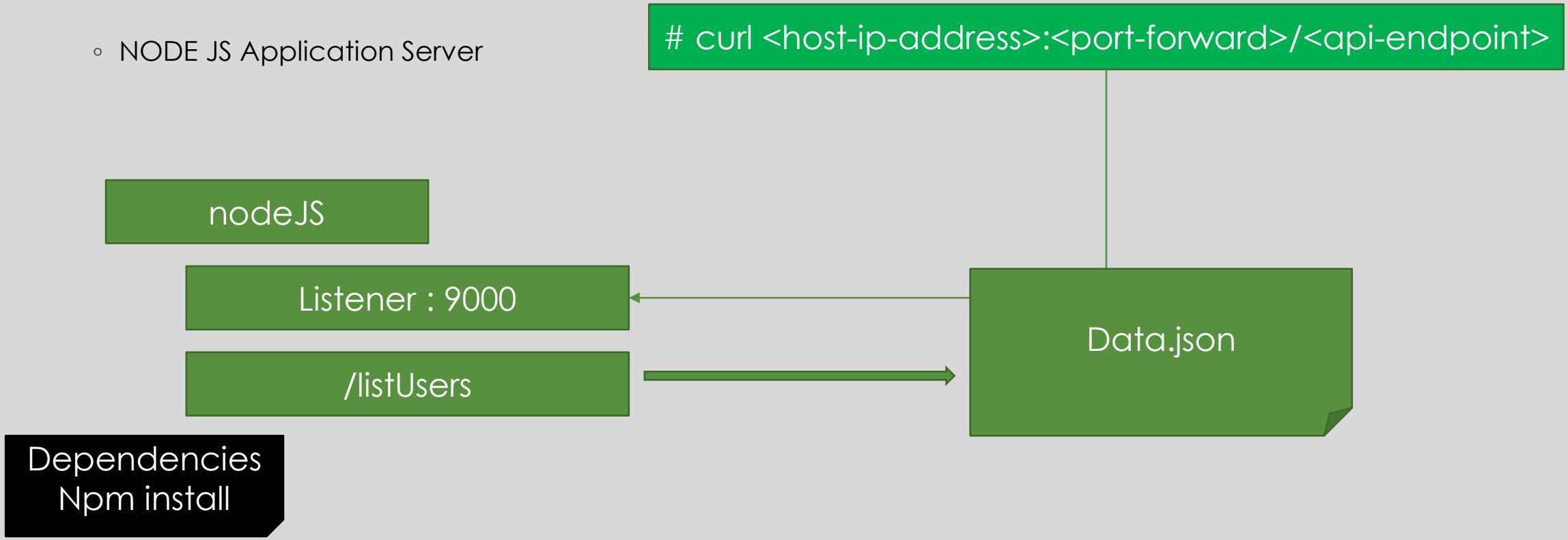


Types of Forwarding

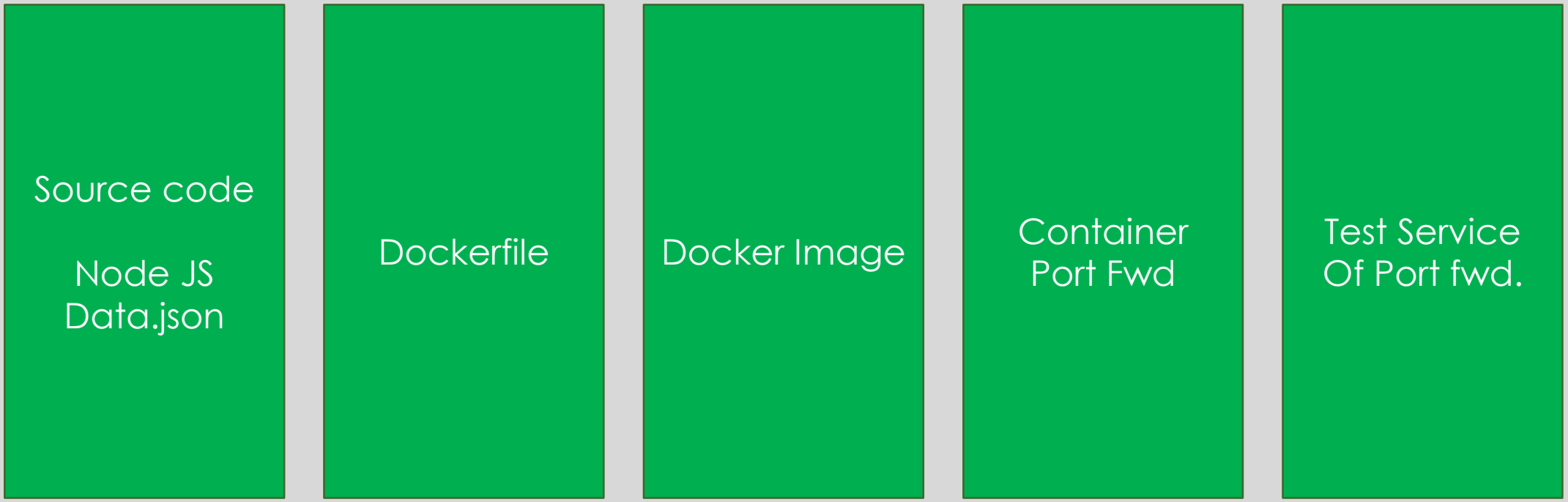
Static Port Forward	Dynamic Port Forwarding
80/TCP : X /TCP	80/TCP → X/TCP
Service via X/TCP	Service via X/TCP
X decided by Ops (Team)	X decided by dockerd
X static for the container	Free port at that point of time (X – moving) start/stop of container
Internal docker router	Service Name (Service Orchestrator) Kubernetes
Port < 30000	> 30000 (32768 – 35999)
-p HP(X): CP (80) –p HP(X1): CP(100)	-P

Use case : Build REST API

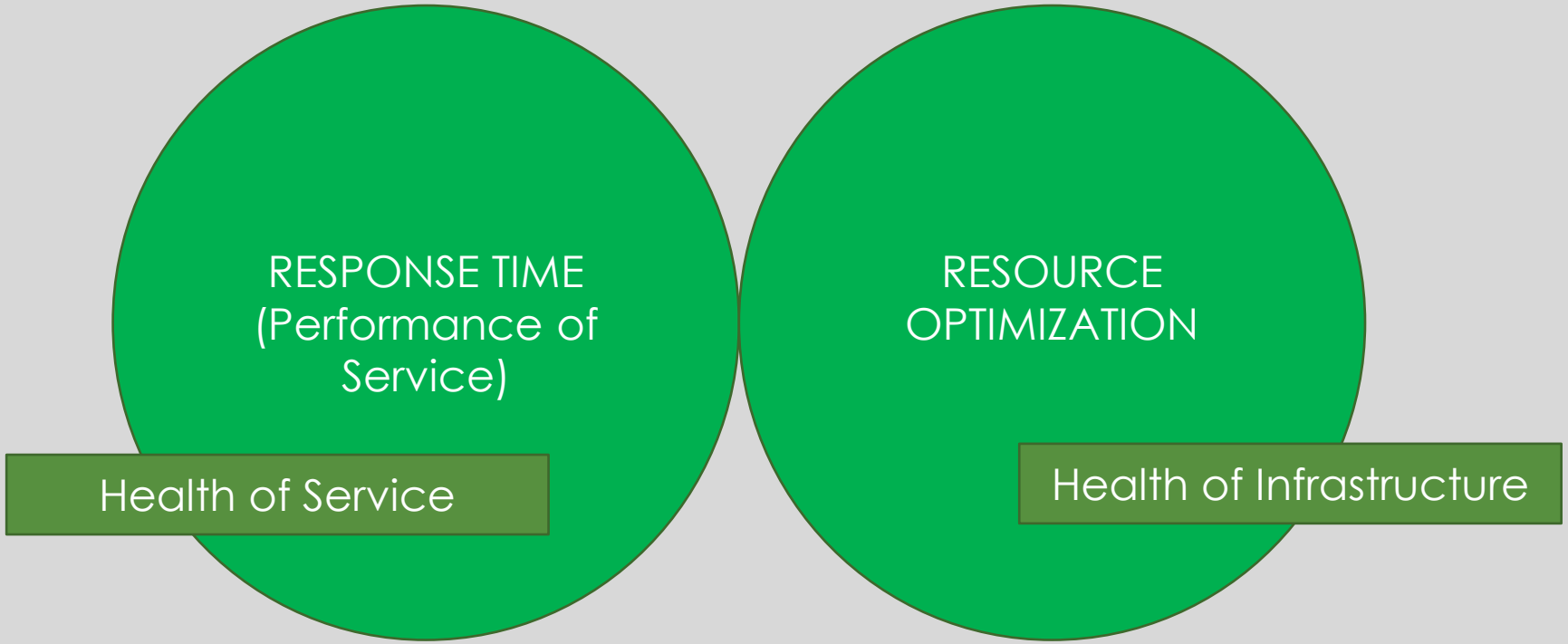
- NODE JS Application Server



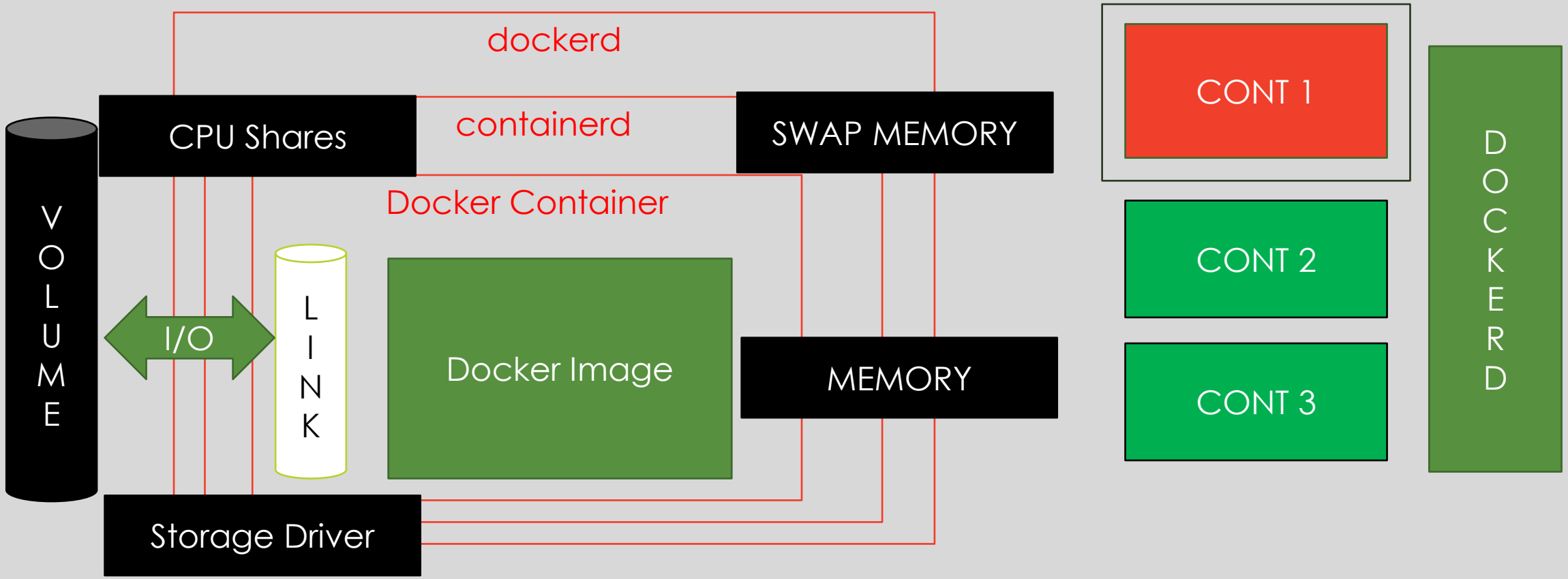
Implement Use Case



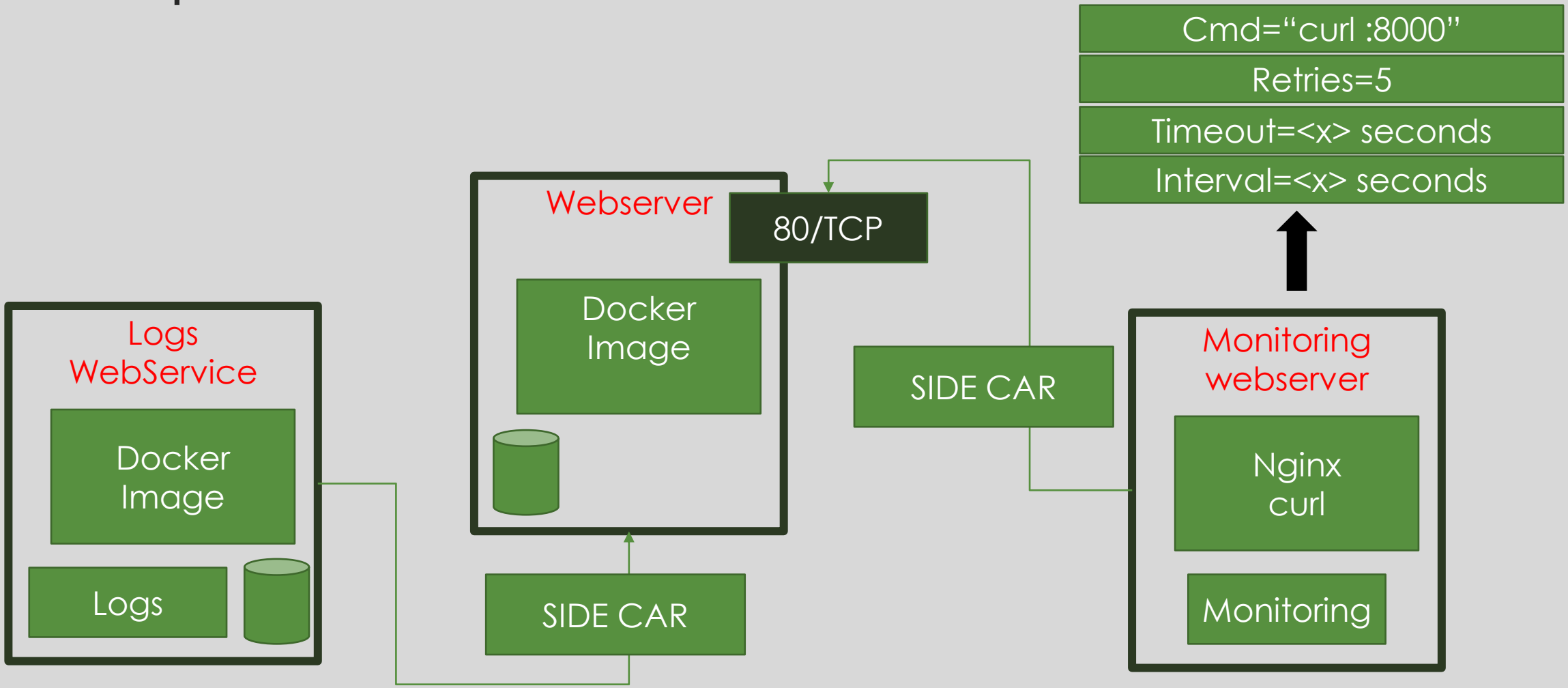
Performance of container



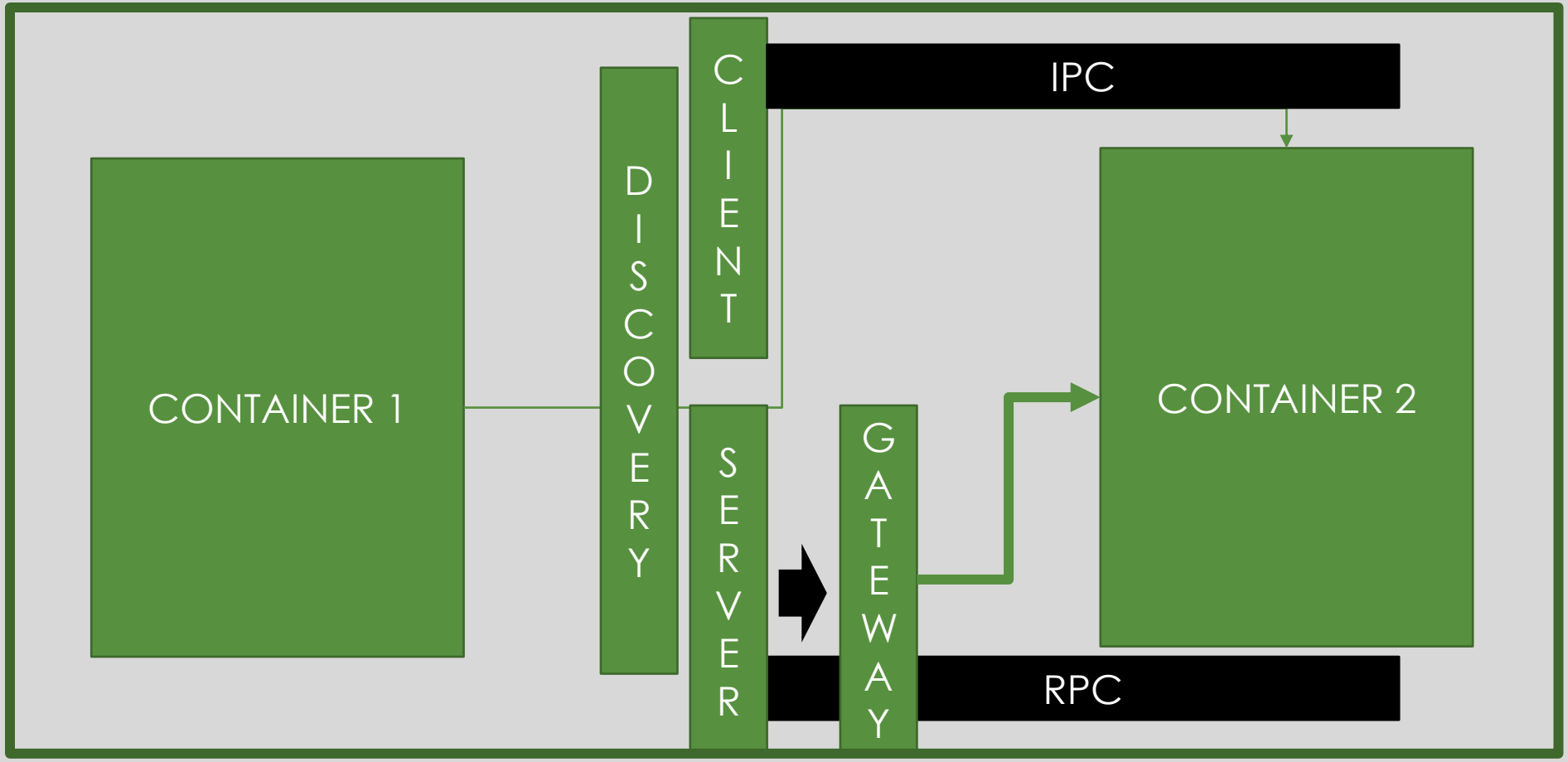
Infrastructure of the container



Response time of container…



Network as configuration



NETWORK configuration

SCOPE OF NETWORK (DRIVER)

BRIDGE (containers with one) docker0

OVERLAY/OVERLAY 2 (multiple dockerd)

MACVLAN (multiple domains)

HOST/ NONE (deprecated)

RUNNING

ESTIMATE CAPACITY THRESHOLD

CIDR (class in domain range)
172.17,0,0- 172.17.255.255
172.17.0.0/16

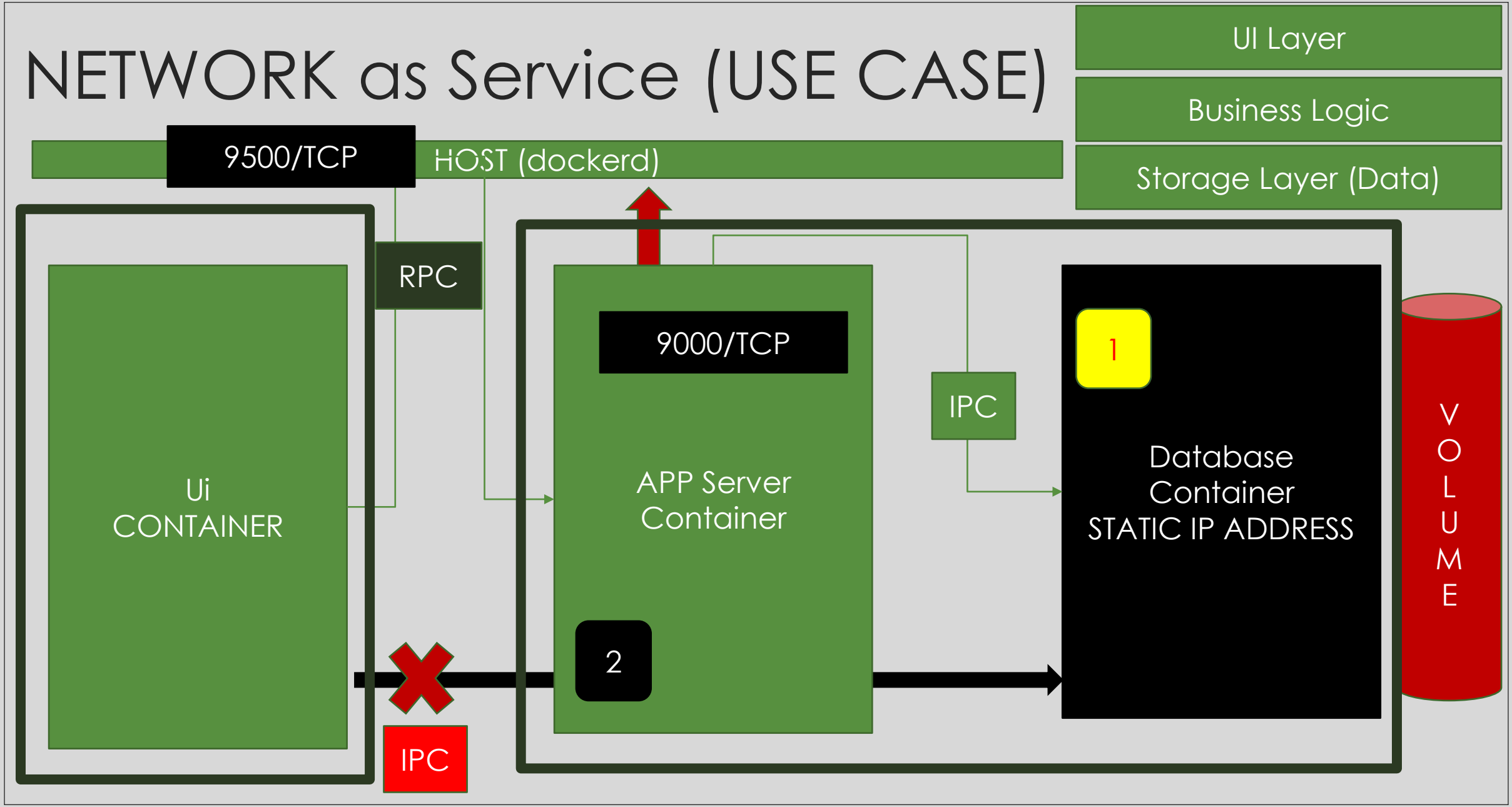
6
5
5
3
3

172.17.0.0 – NET MASK

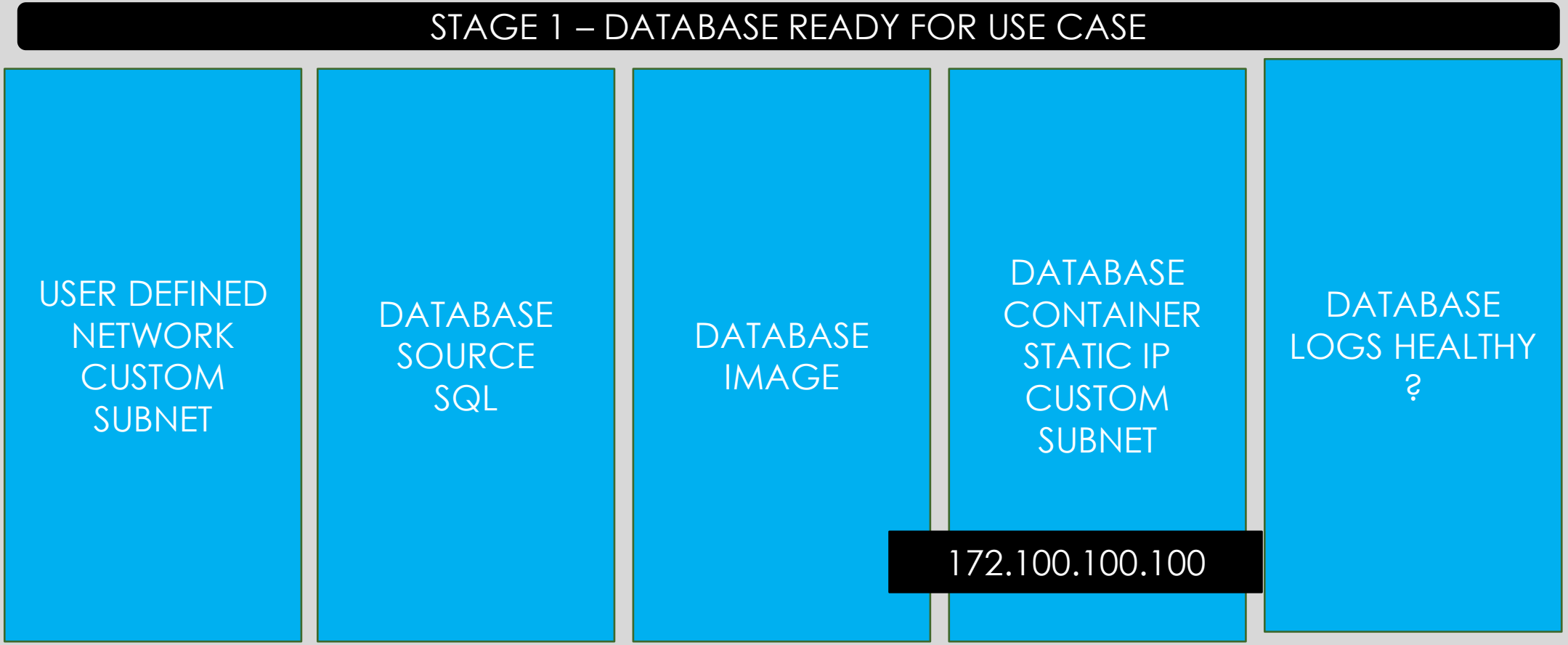
172.17.255.255 – NET MASK(2)

172,17.0.1 -- GATEWAY

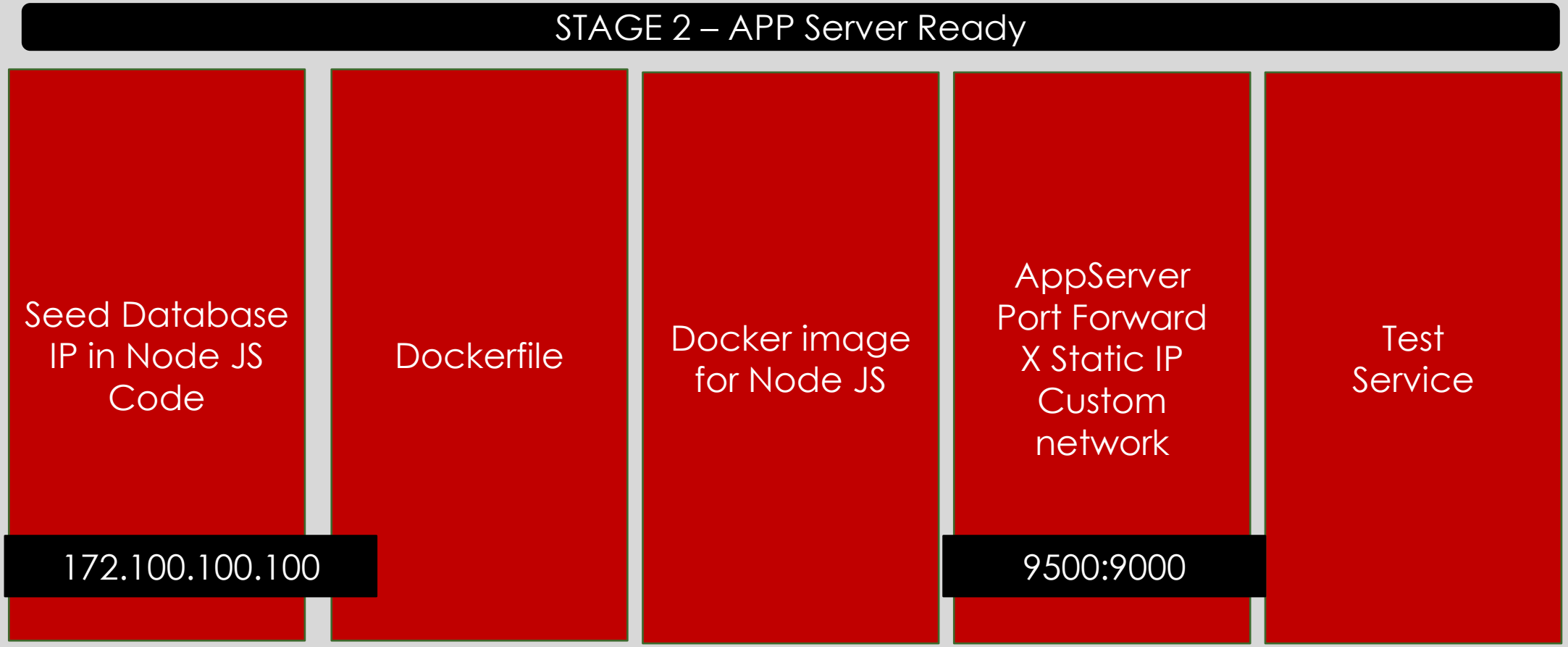
NETWORK as Service (USE CASE)



Steps to implement Use case (Slide 60)



Steps – contd.. (slide 61)



Day 4

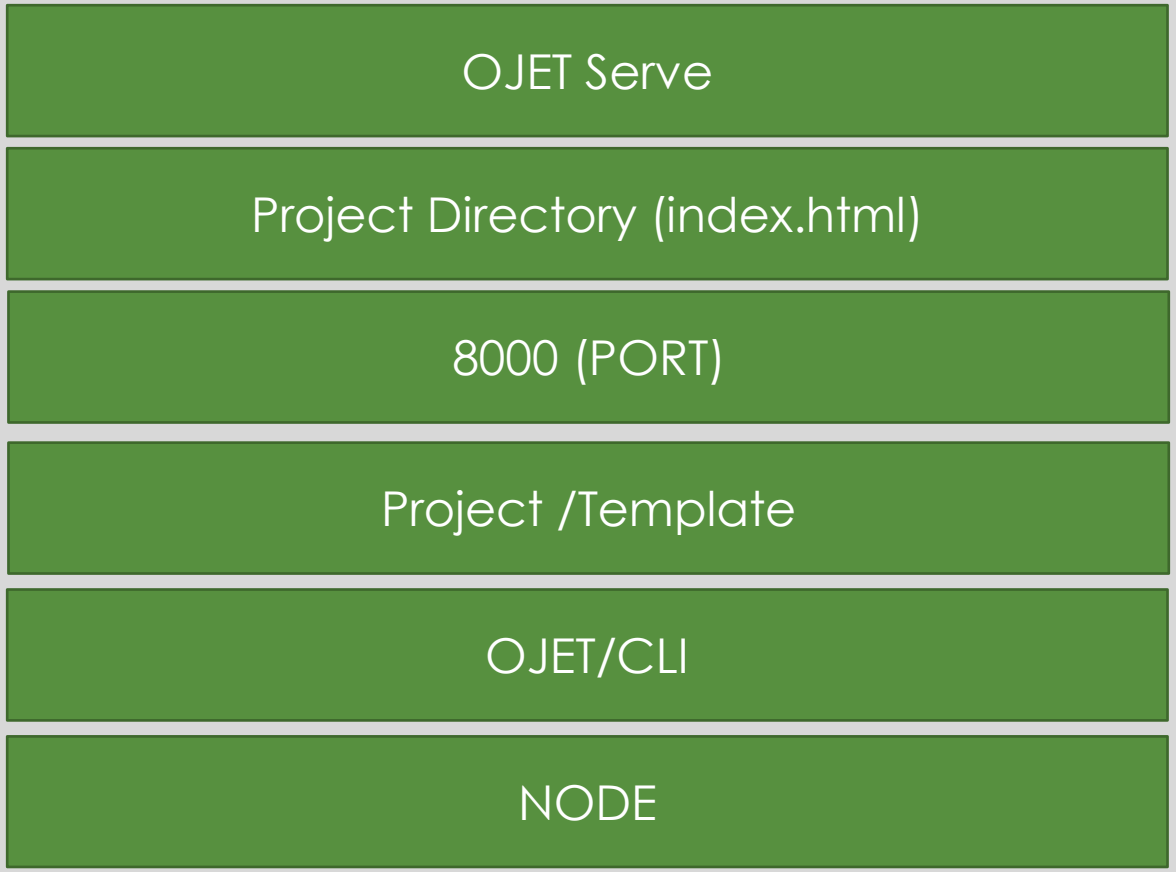
- Trouble shooting (Log Policies)
- UI OJET Container…
- Container policies (Transient)
- Release management
- Release images
- Kubernetes Architecture
- Kubernetes Installation
- Kubernetes Pods
- Kubernetes Objects
- Service management (k8s)

Log configuration properties..

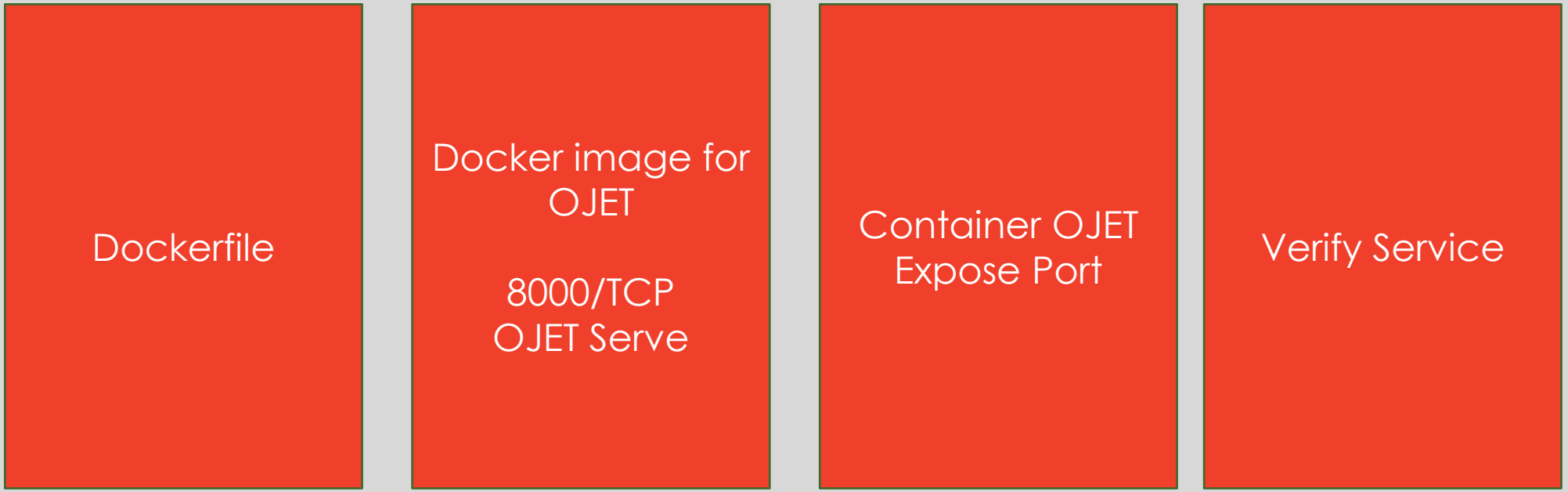
# Log Files	Log Buffer Size	Log Mode	Log Mode
1 log file	4M	**Blocking** _once log file reaches log size – no more logs written	JSON-FILE FLUENTD (EFK) SYSLOG SPLUNK TextLogs Log4J Awslogs/gclogs None
32767 log files	K to G to T	Non-blocking : -- cyclic logs , logs are rewritten , new logs as assurance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

UI

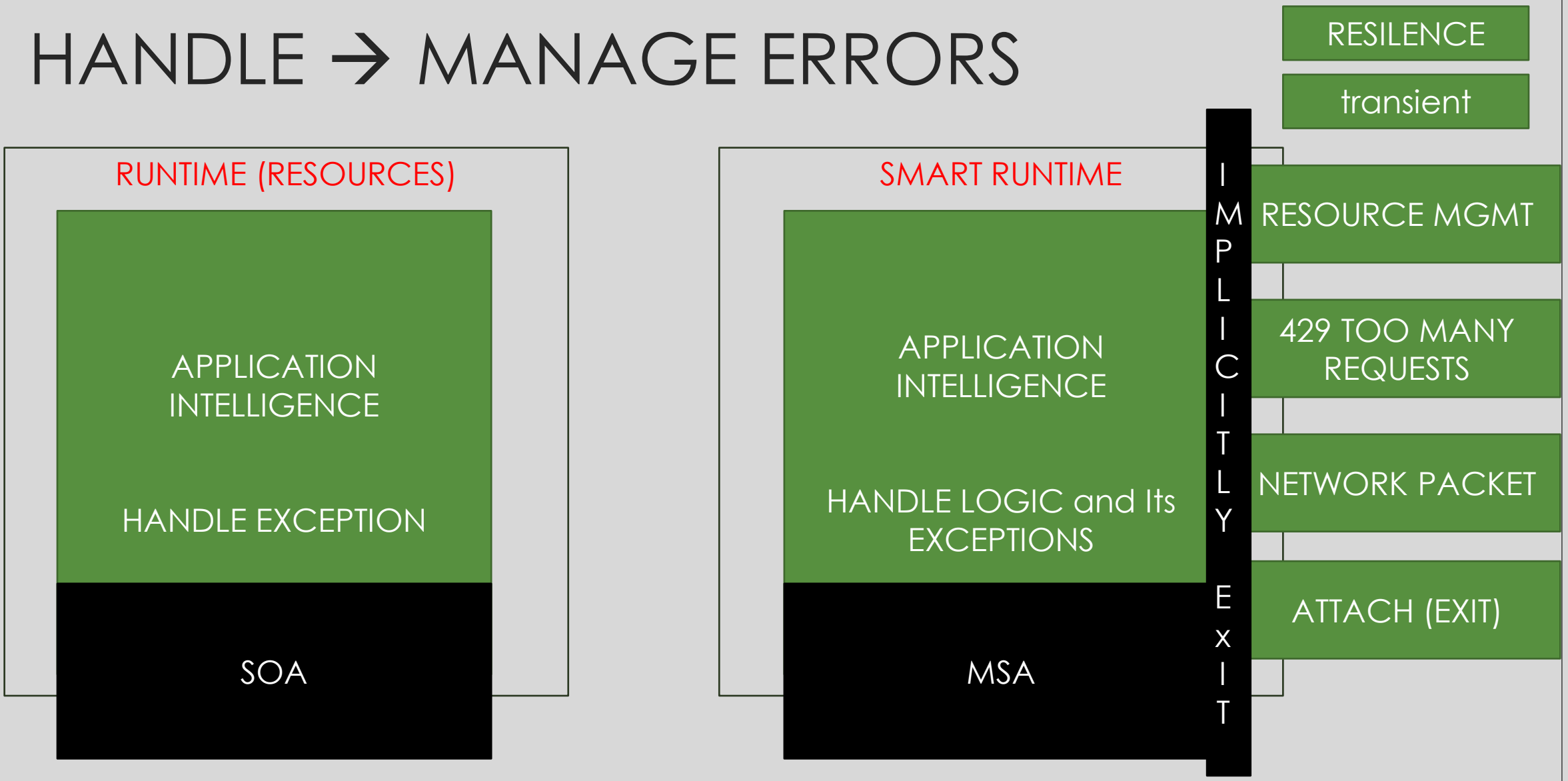
- Open responsive design
- OJET – HTML CSS , Require.JS , Knockout,jS (data binding)
- Node JS



Implement UI



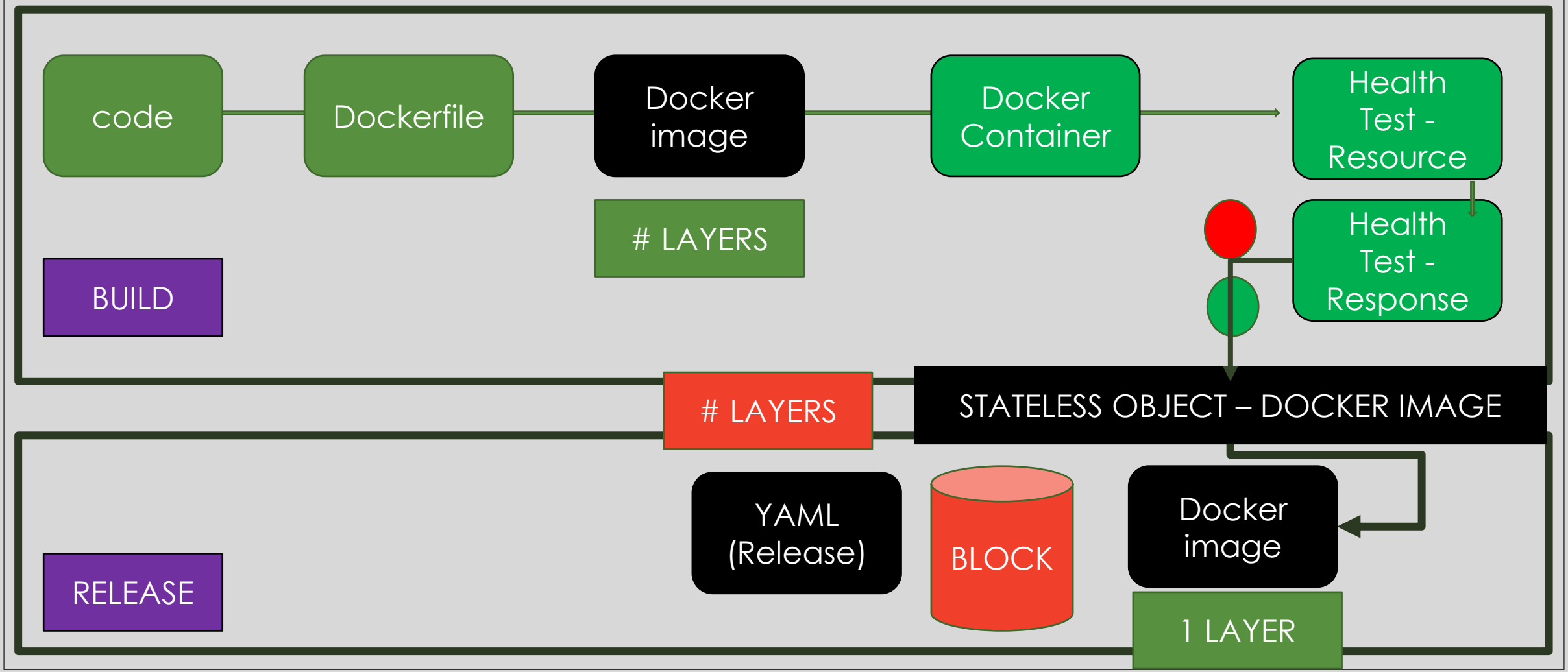
HANDLE → MANAGE ERRORS



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”
```

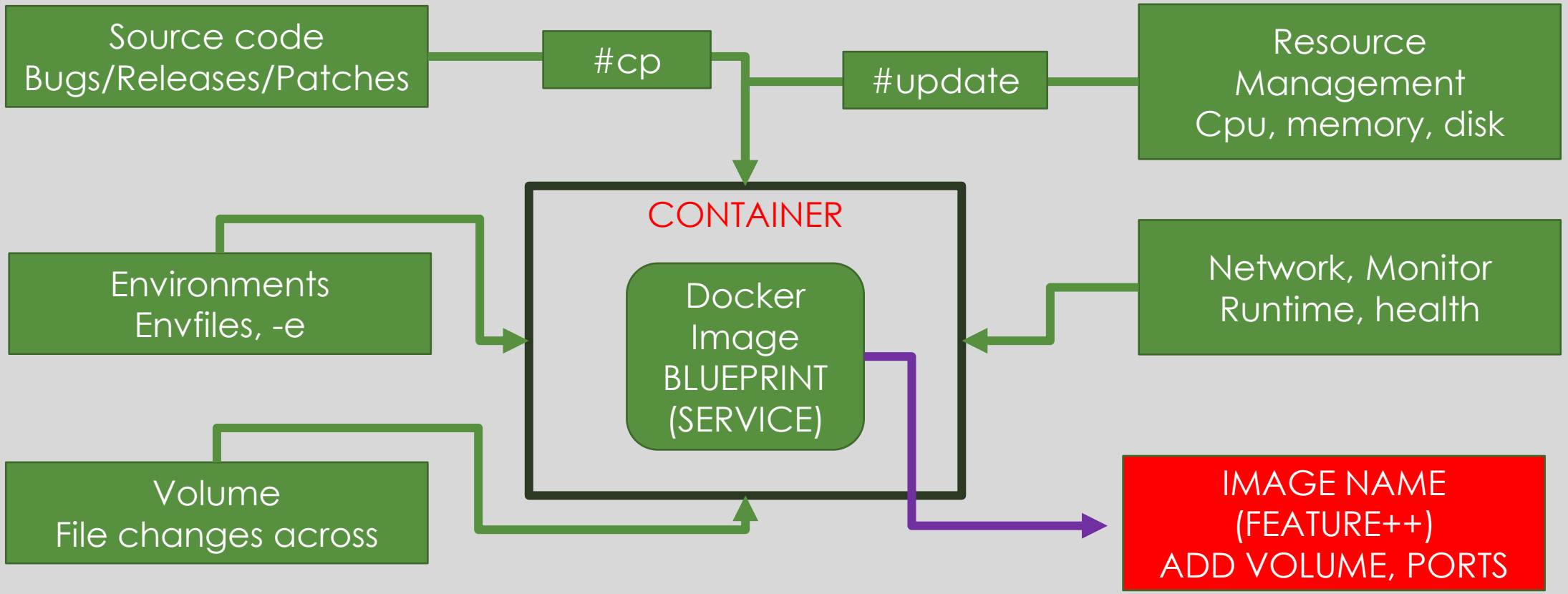
Compose version

Meta data for container

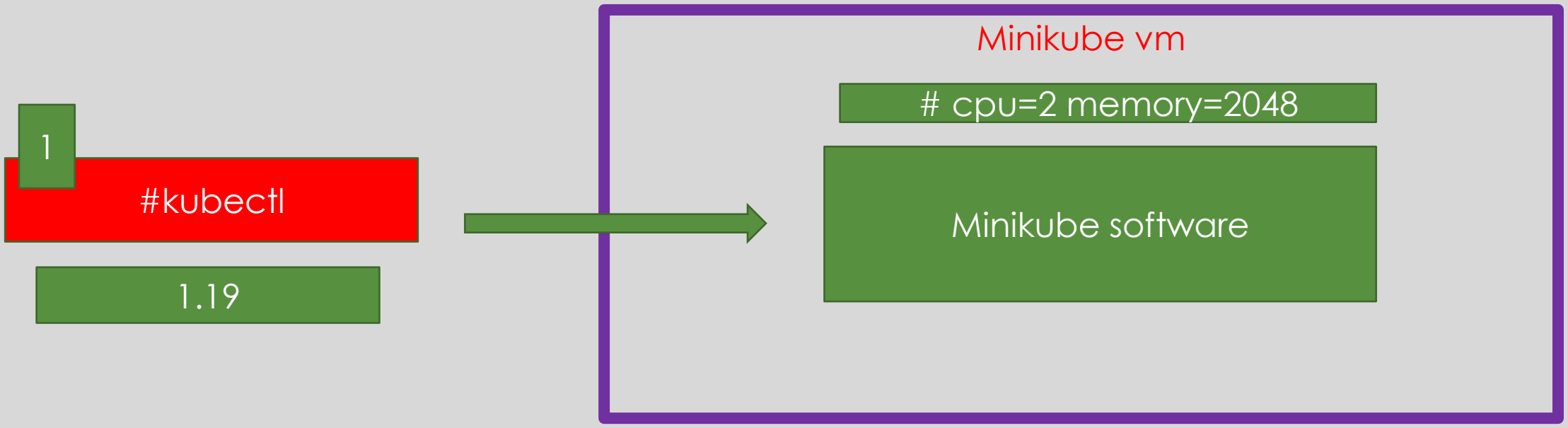
Meta data for container 2

Static Port forwarding of container

Minimizing changes…



Environment for K8s



Use case(s)

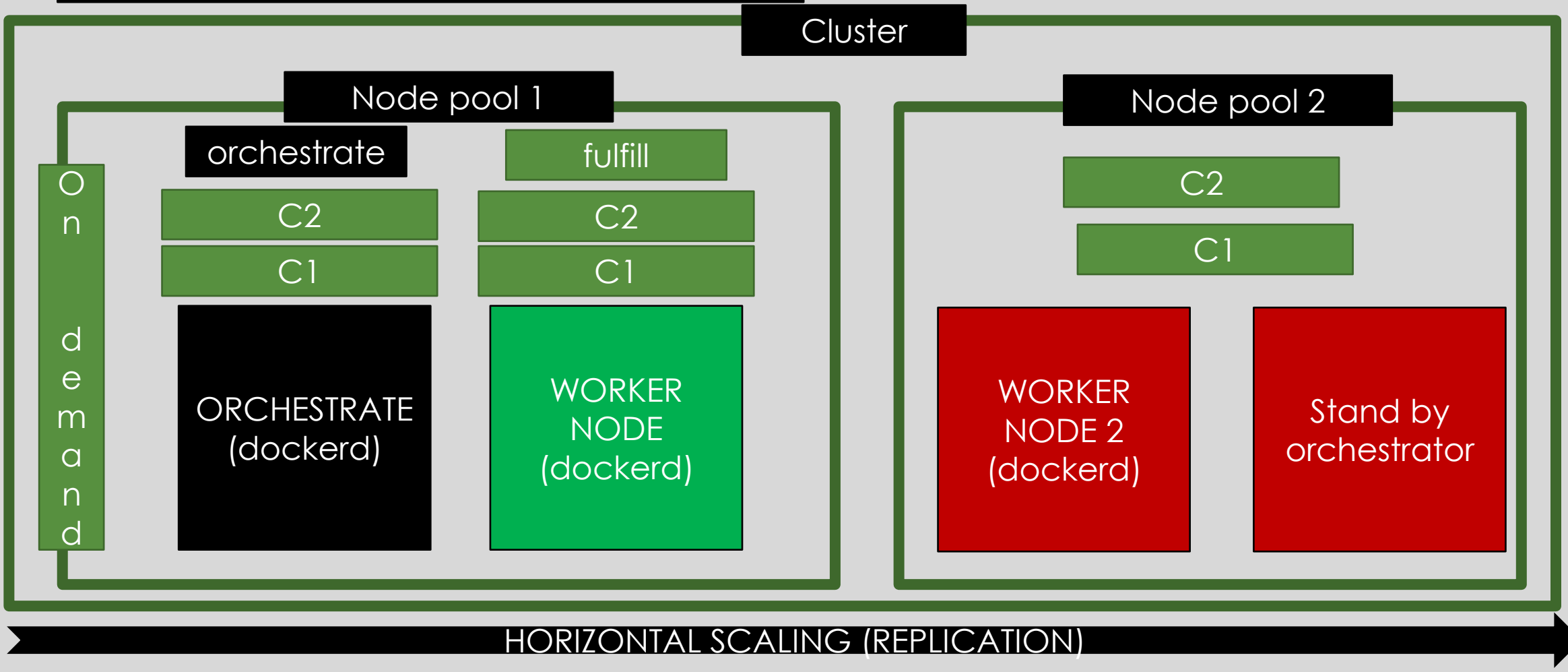
Why to use Docker	Why to use K8s
+ Applications are Portable, Runtime are not portable. (Runtimes –Application changes) [PORTABILITY] + Performance of Application + Option to Scale as a Service	+ Orchestrate Services as Containers (heal) + Horizontal (machines), Vertical (Containers) + Rollout and Release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

Horizontal scaling

Node=machine

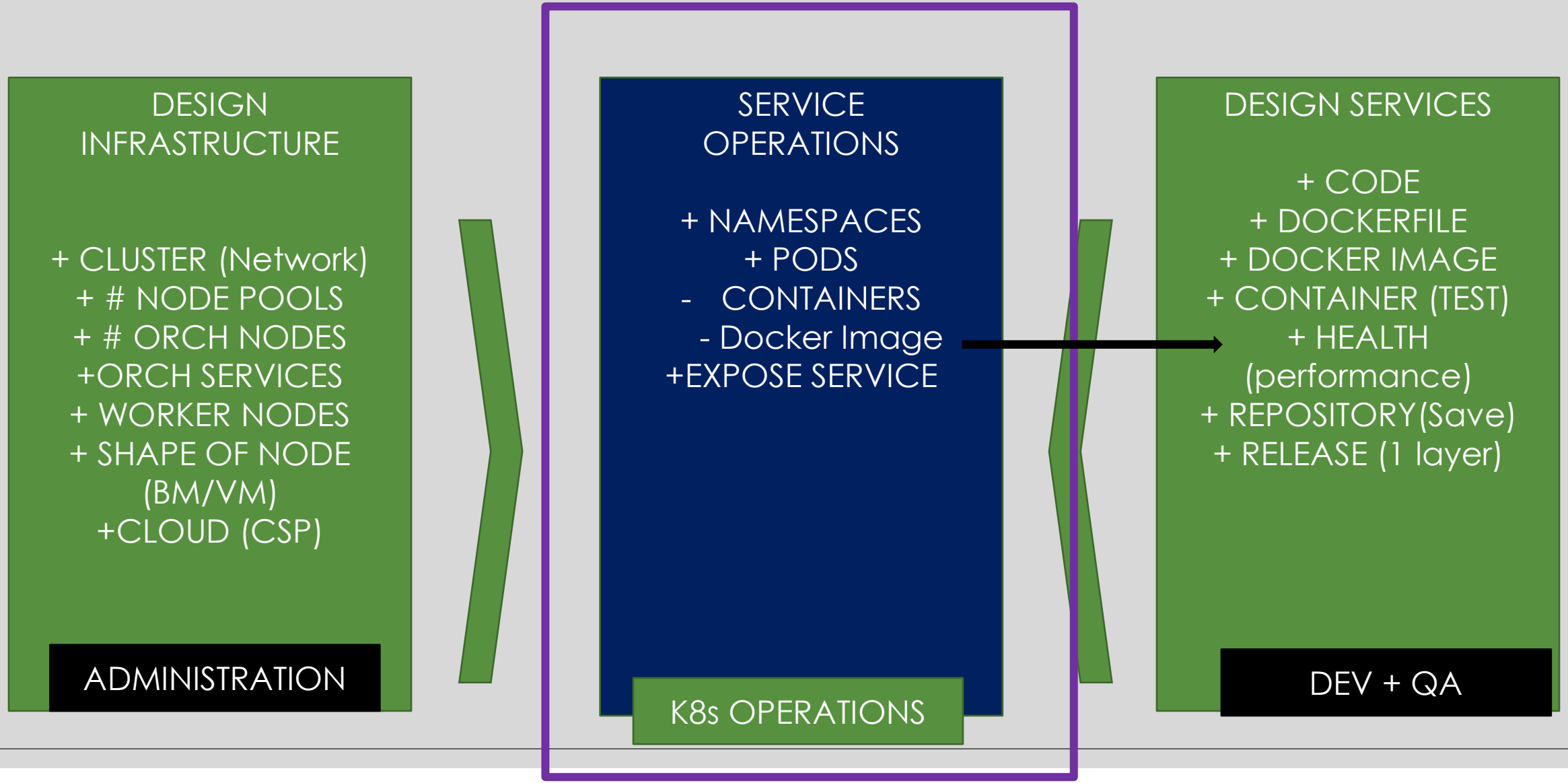
VM/BM = $

Monitor,Load Balance, log and fulfill

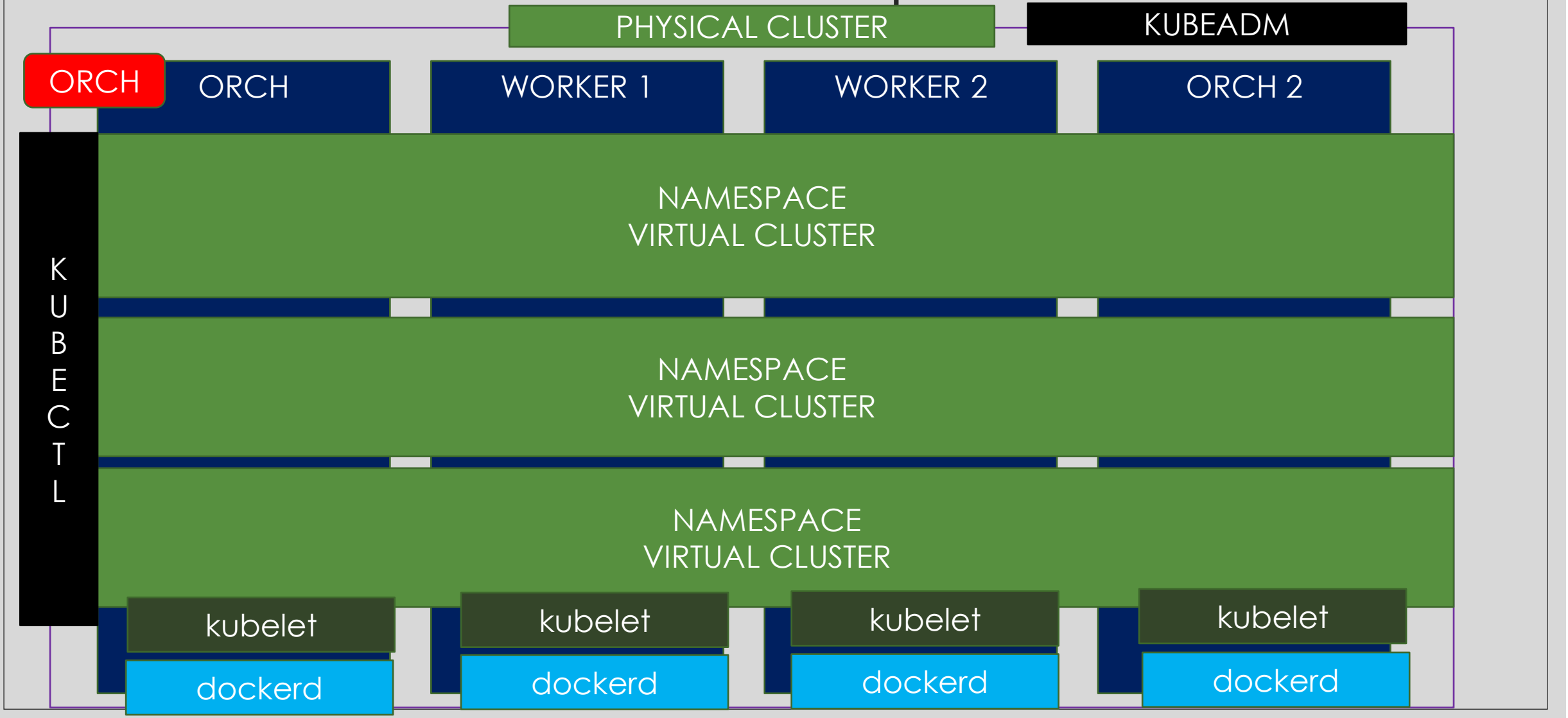


HORIZONTAL SCALING (REPLICATION)

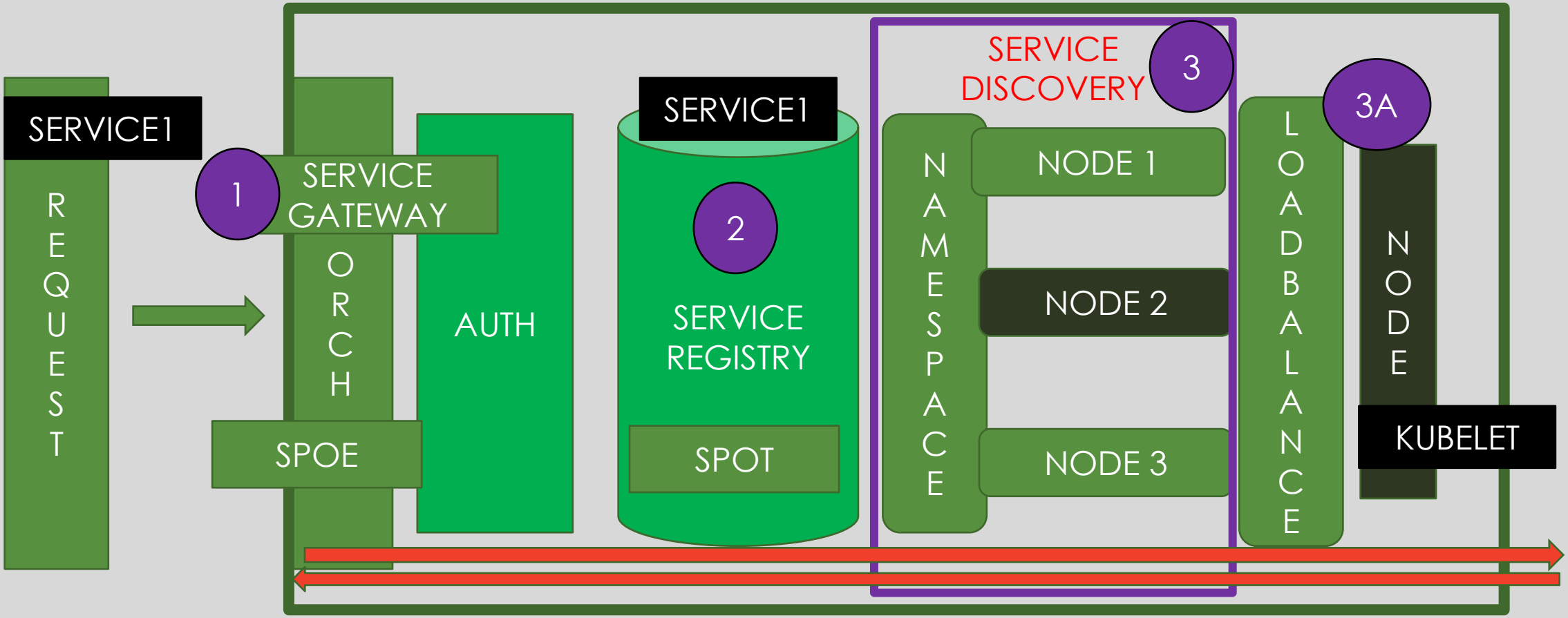
3 roles


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



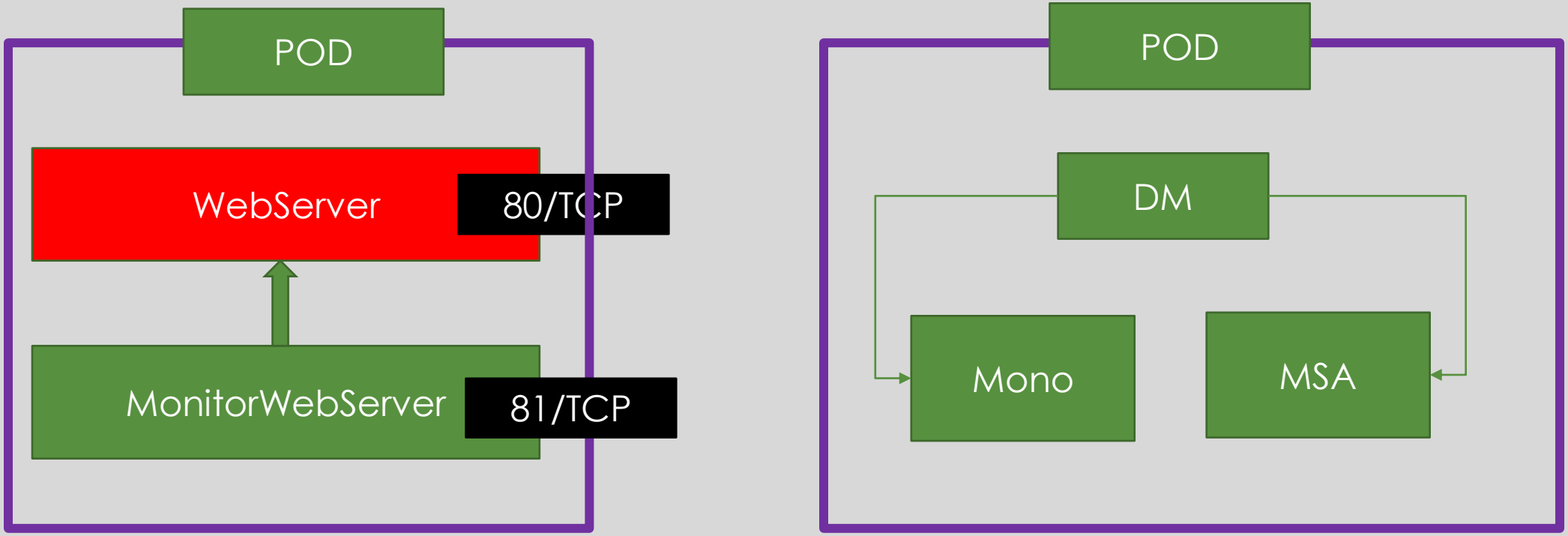
Orchestrator...



Orchestrator tools…

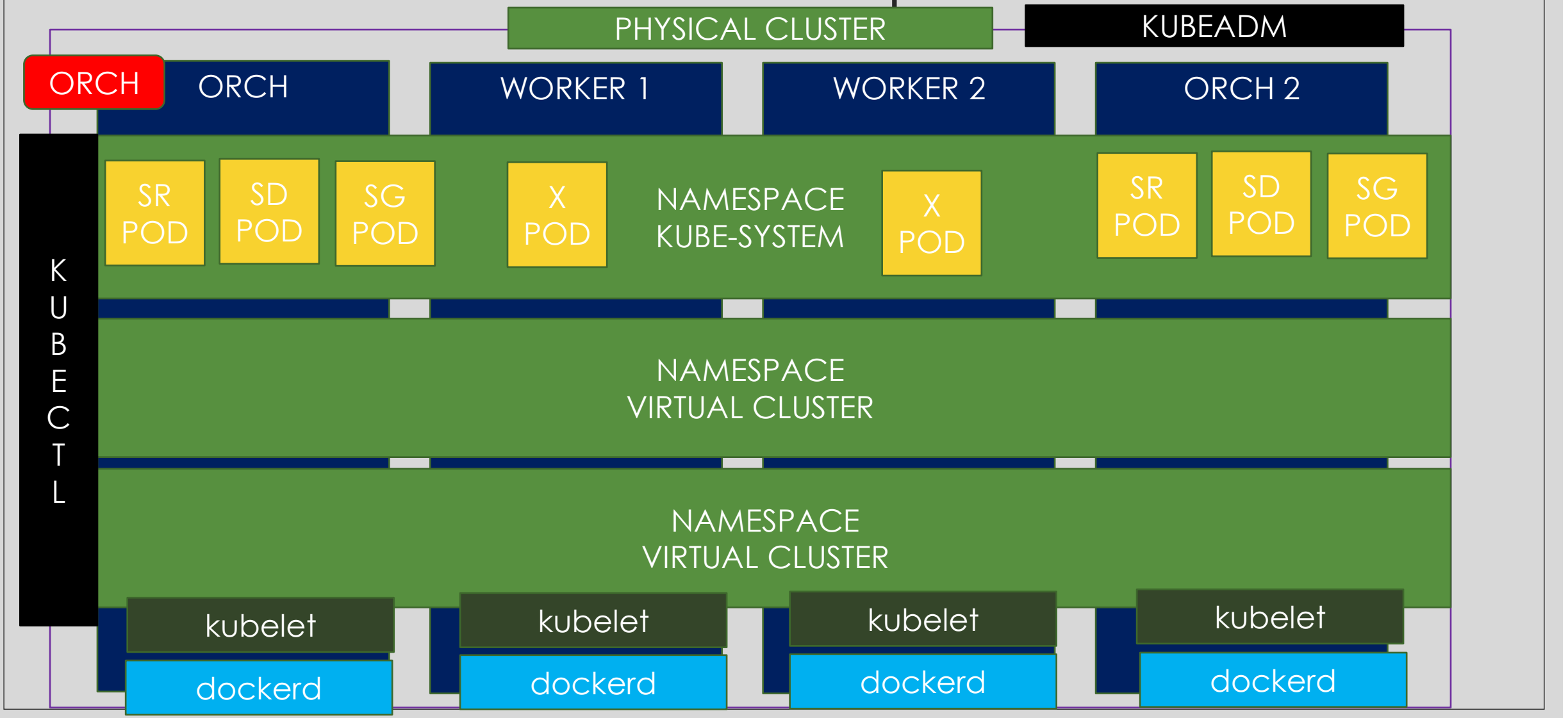
Service Registry/Service Discovery	Service Gateway	Ingress/Egress
Etcd (key value pair) API Server (service discovery)	Kube DNS (core DNS)	Gateway
Kong (oracle,mysql rbdms)	JBOSS , VertX	ISTIO , LinkerD
Zoo Keeper (no sql unstructured)	OTD (oracle traffic director)	Voyager , HA proxy, Traffeik
Consul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

POD

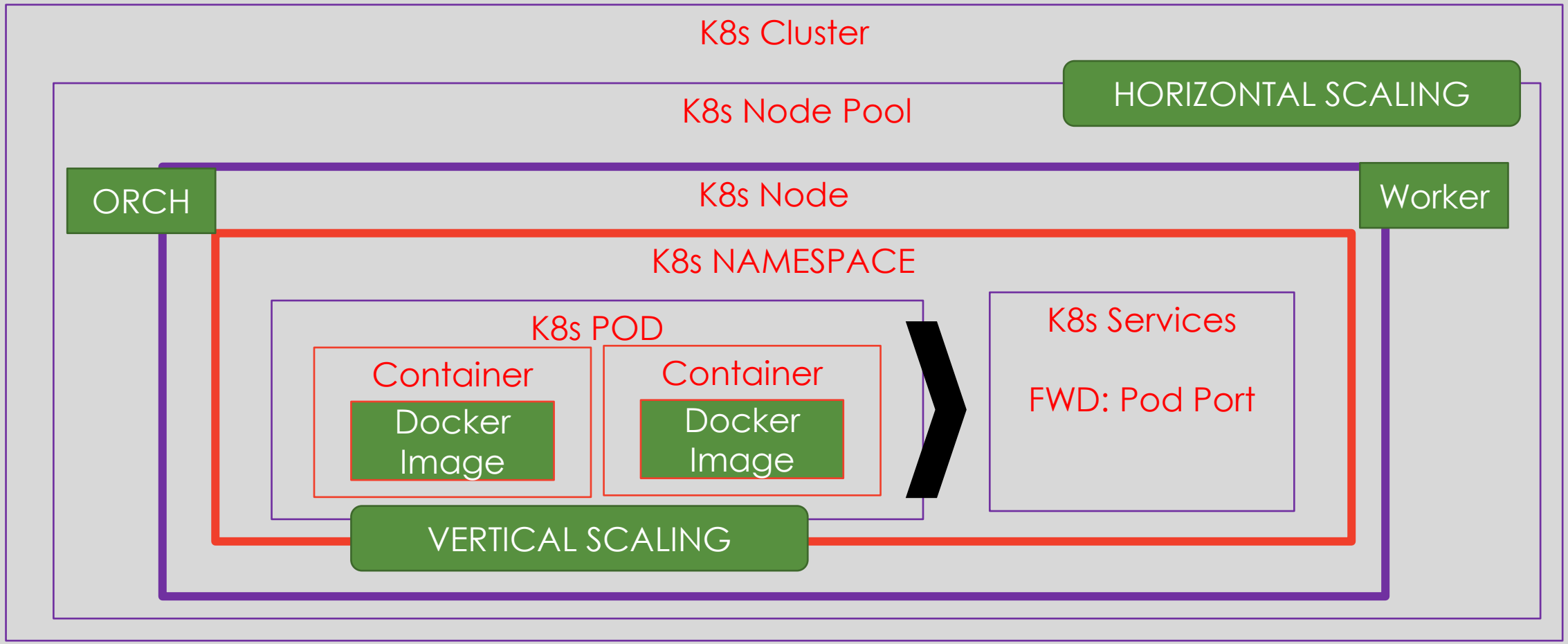


Unit of Abstraction for Scaling one or more containers exposing one or more ports (services)

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

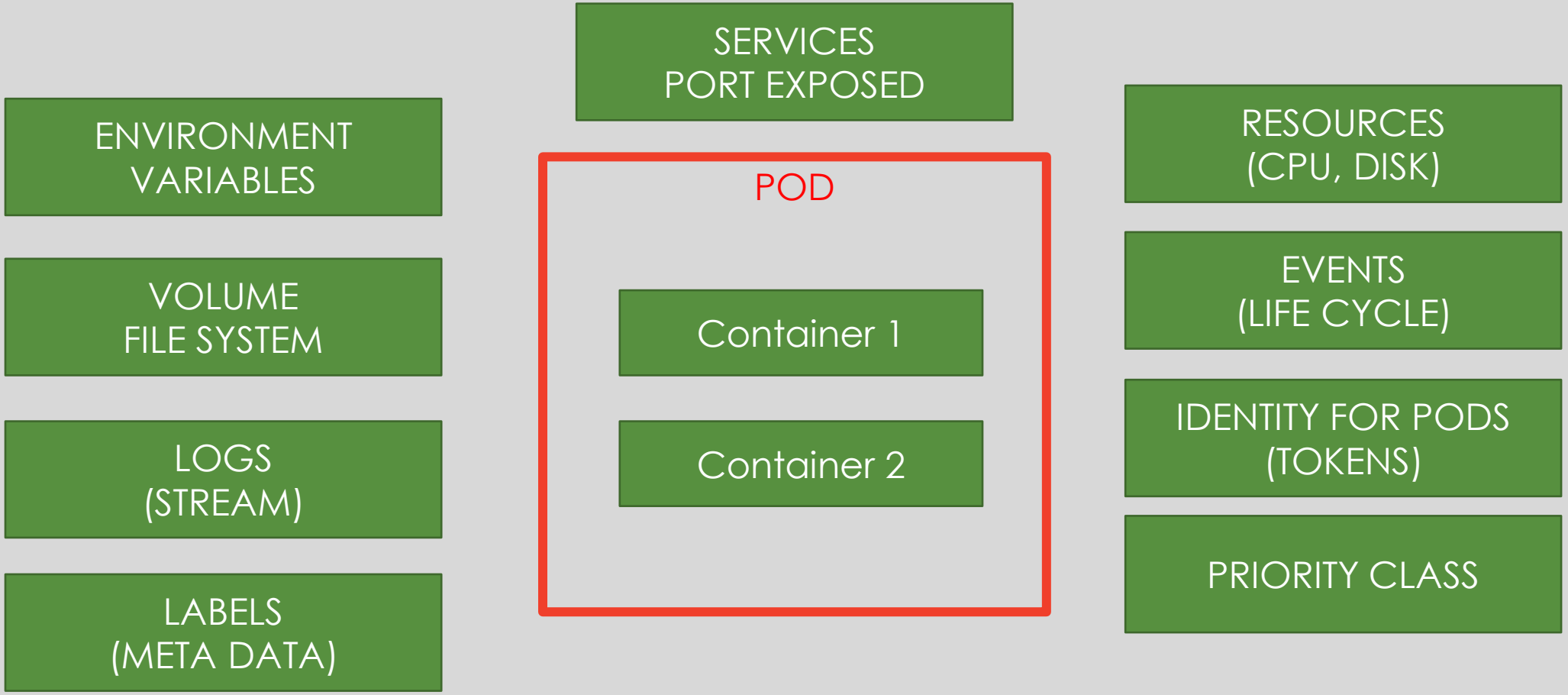
kubectl api-resources

- 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)

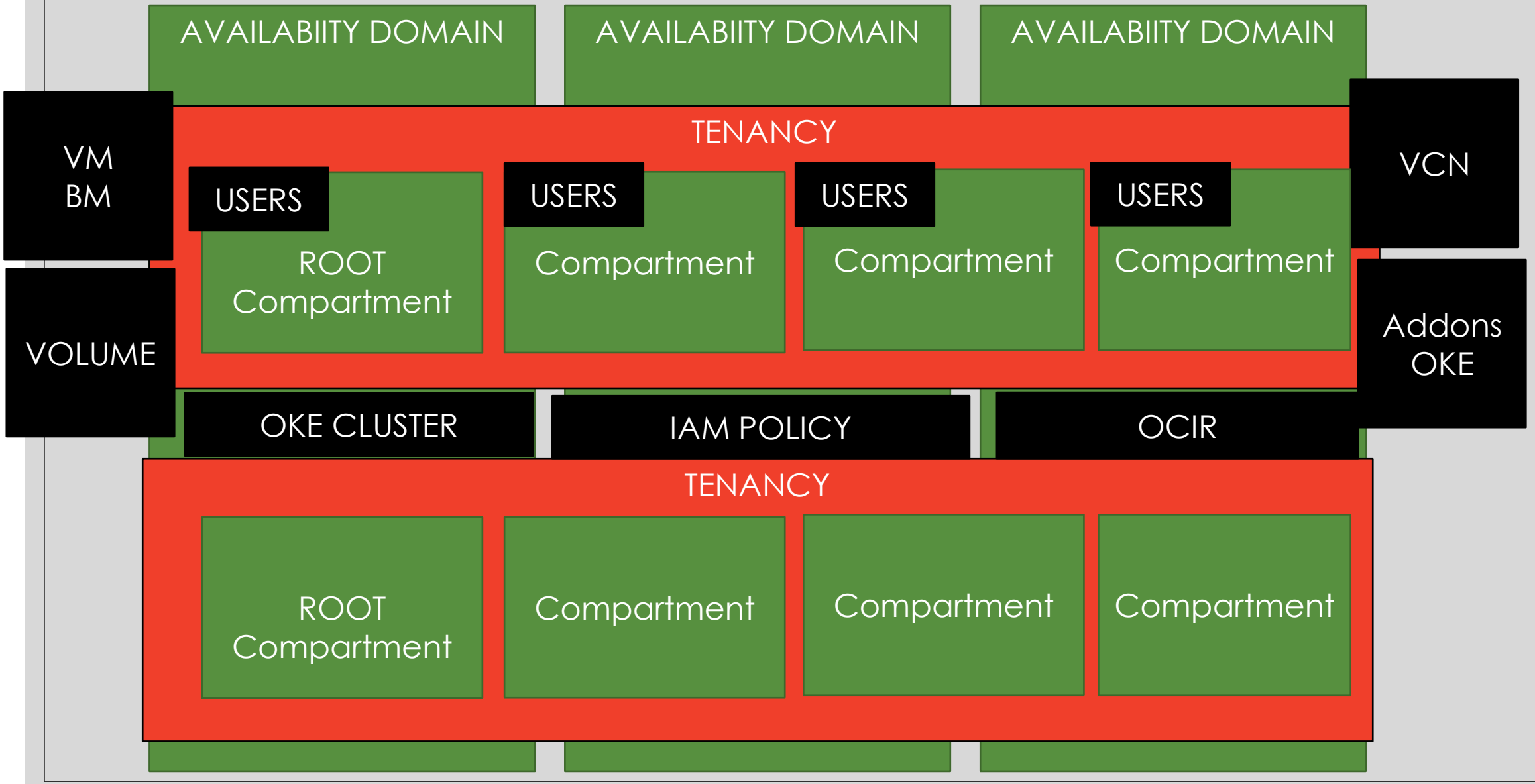
POD properties



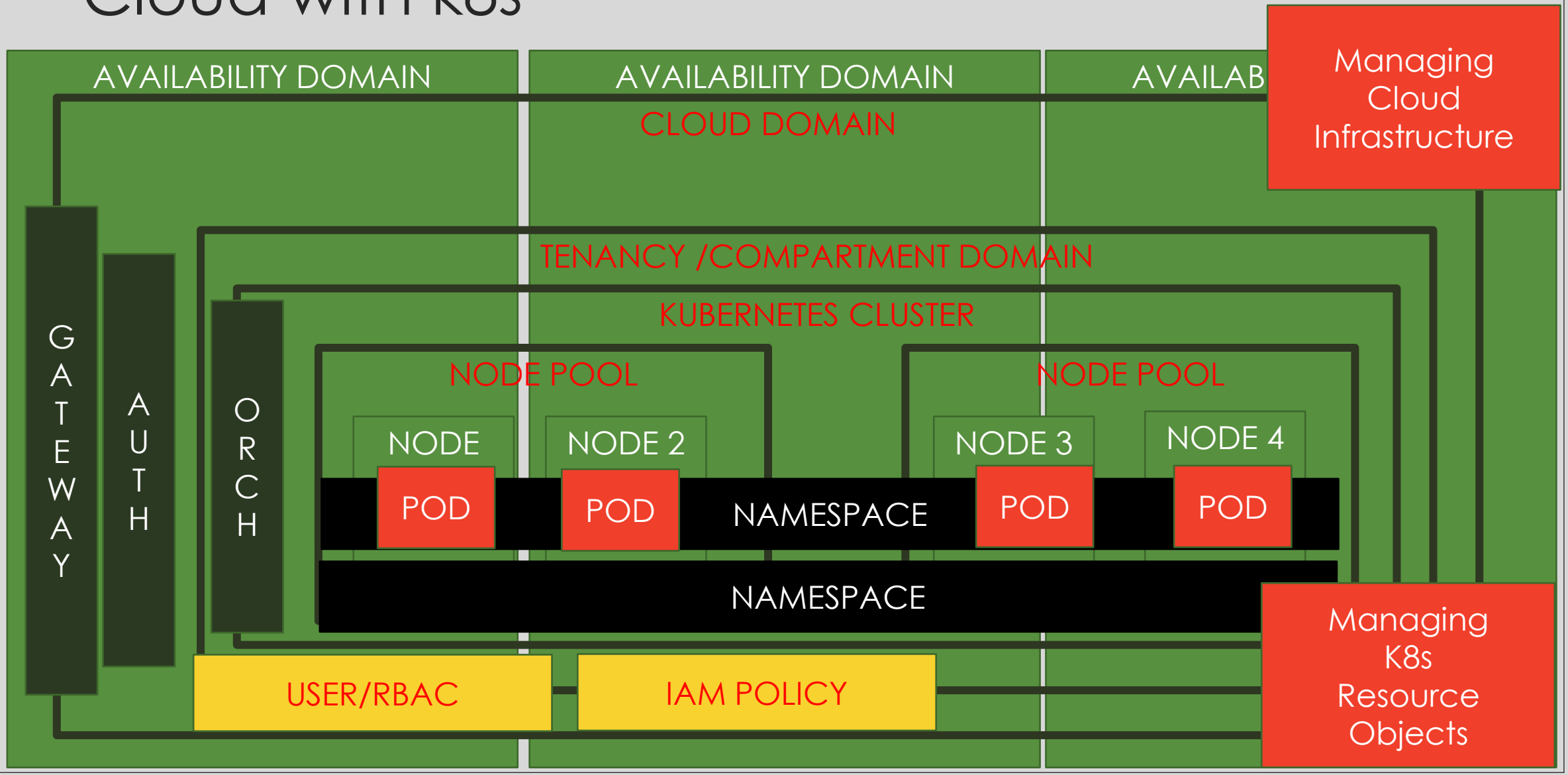
Day 5

- Cloud Architecture
- Cloud Native
- Pod Services
- Service Management
- Environments for POD
- Multi Container POD (side car)
- Replica of PODS
- Use case – Communication between Pods
- Troubleshooting in K8s
- Rollouts #

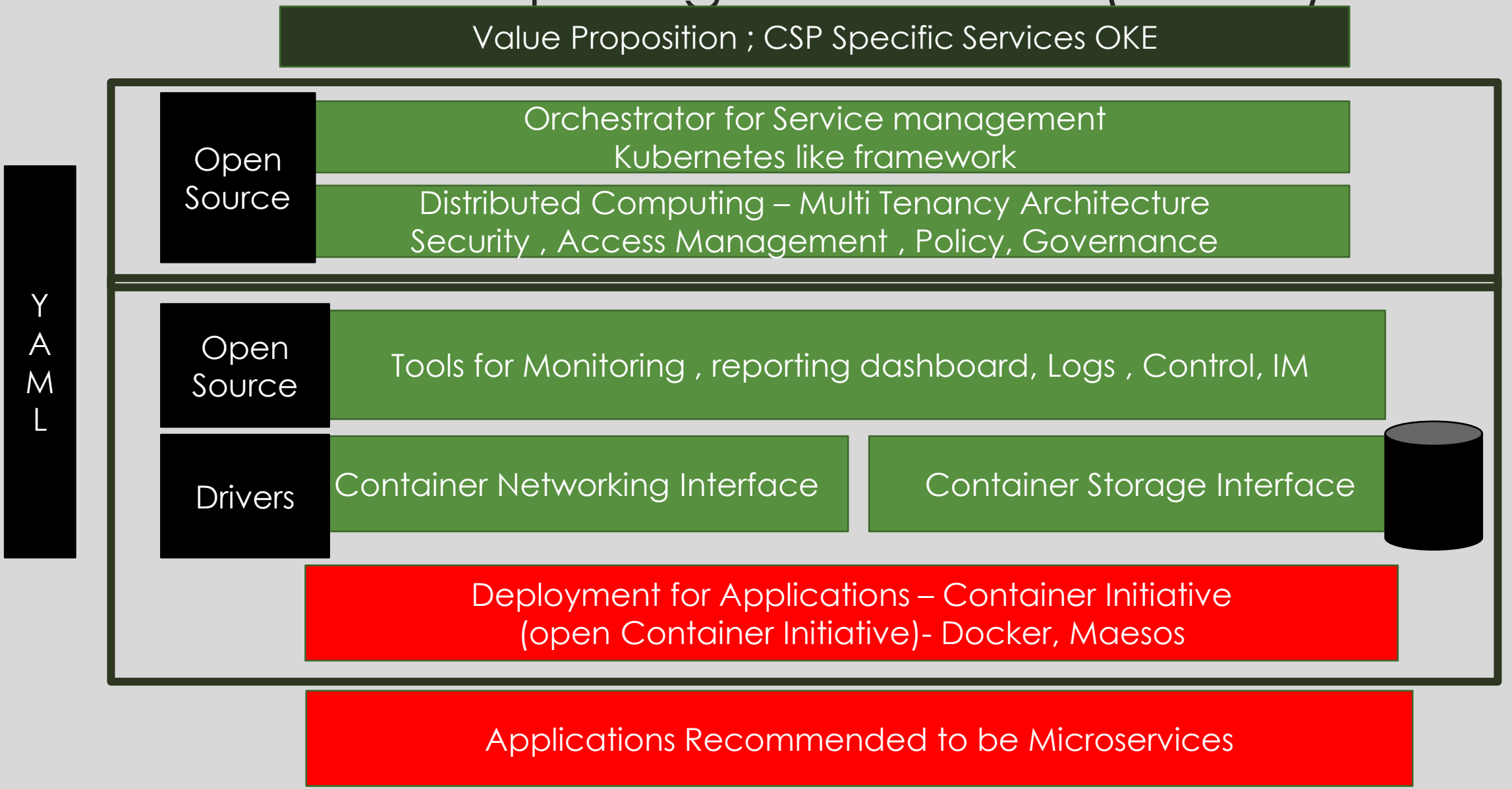
Cloud Architecture



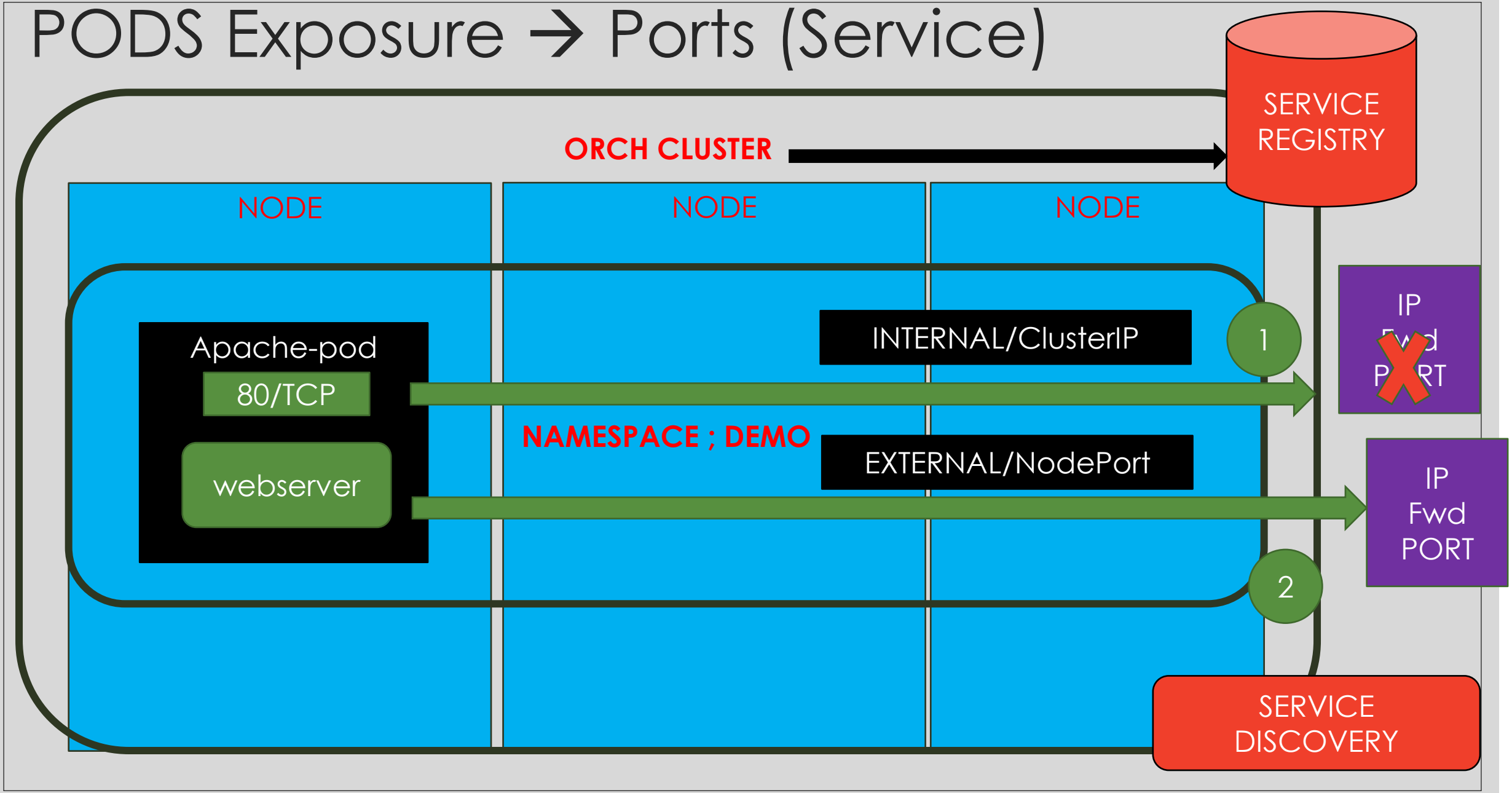
Cloud with k8s



Cloud native computing foundation (CNCF)

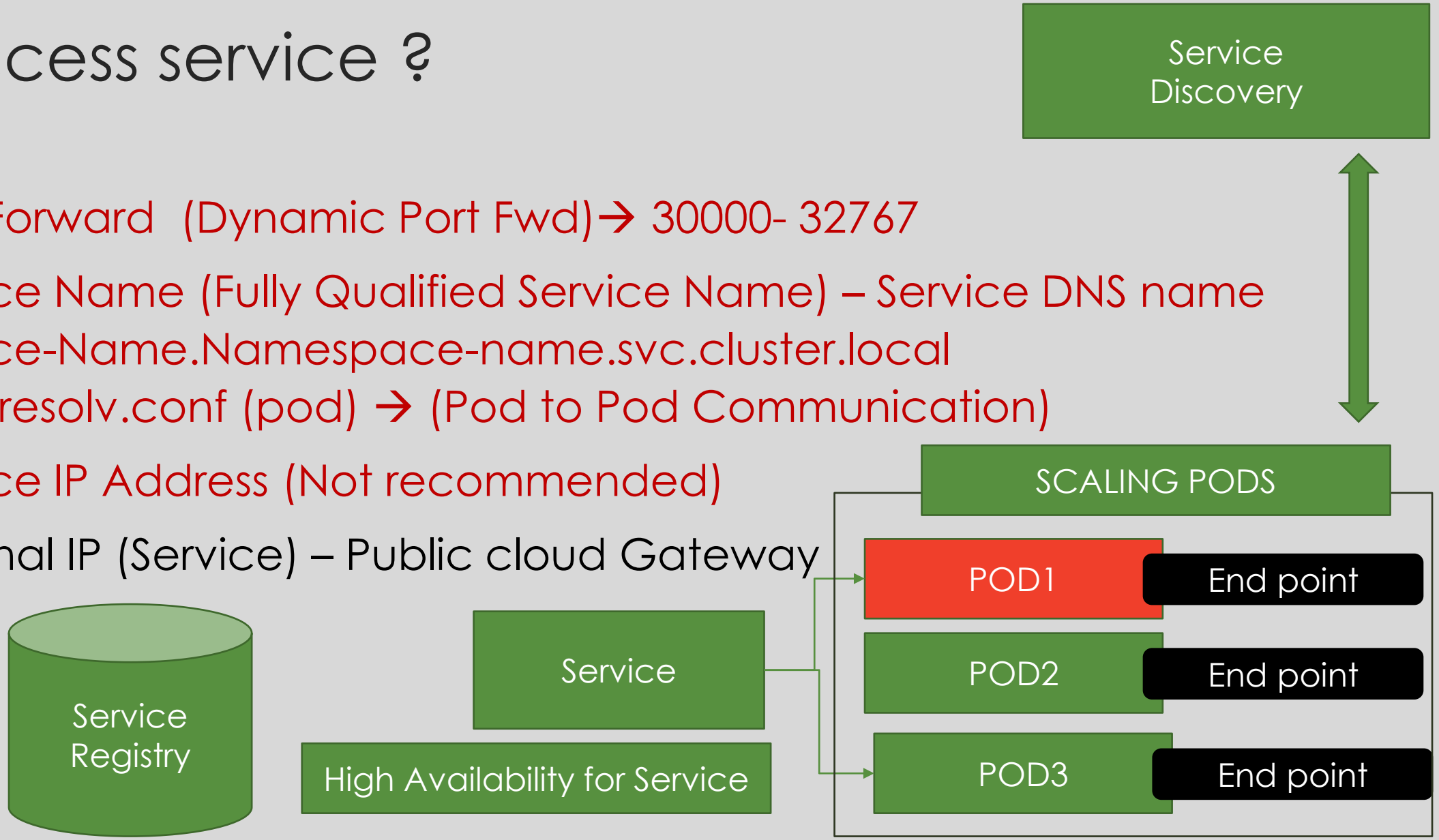


PODS Exposure → Ports (Service)

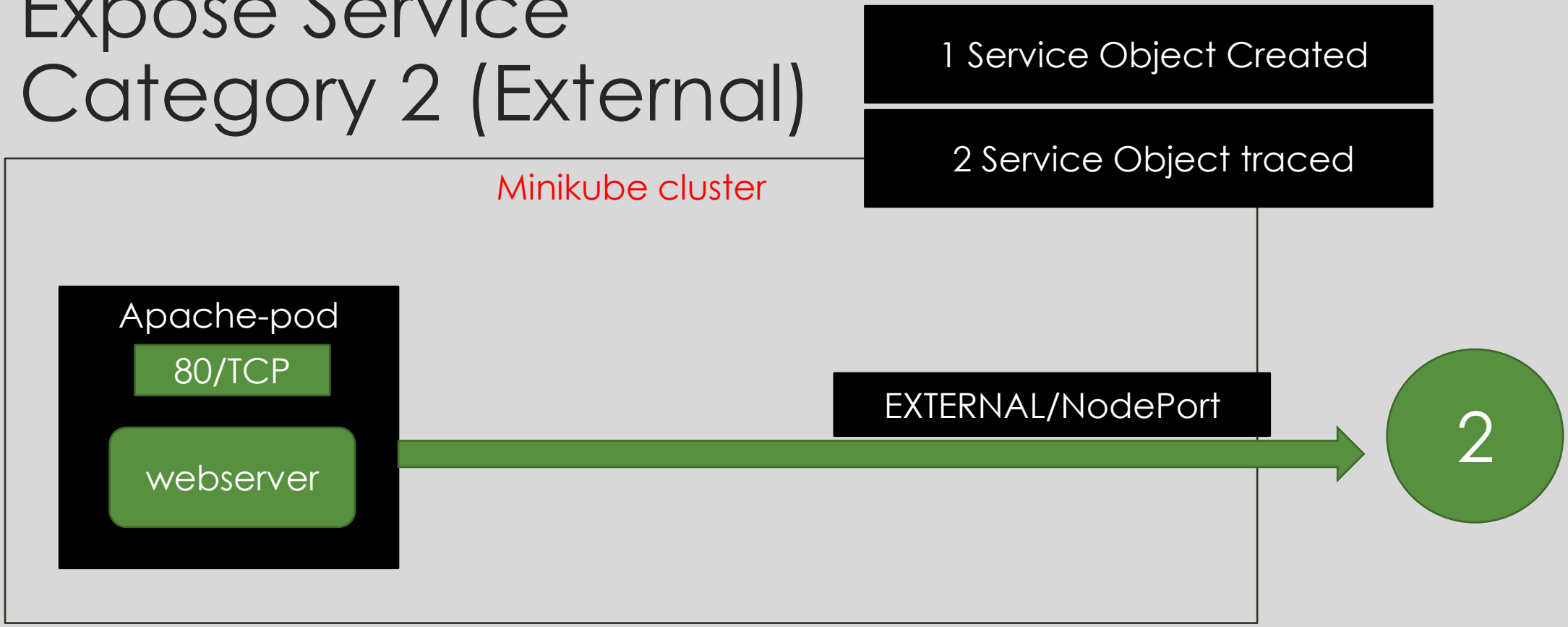


Access service ?

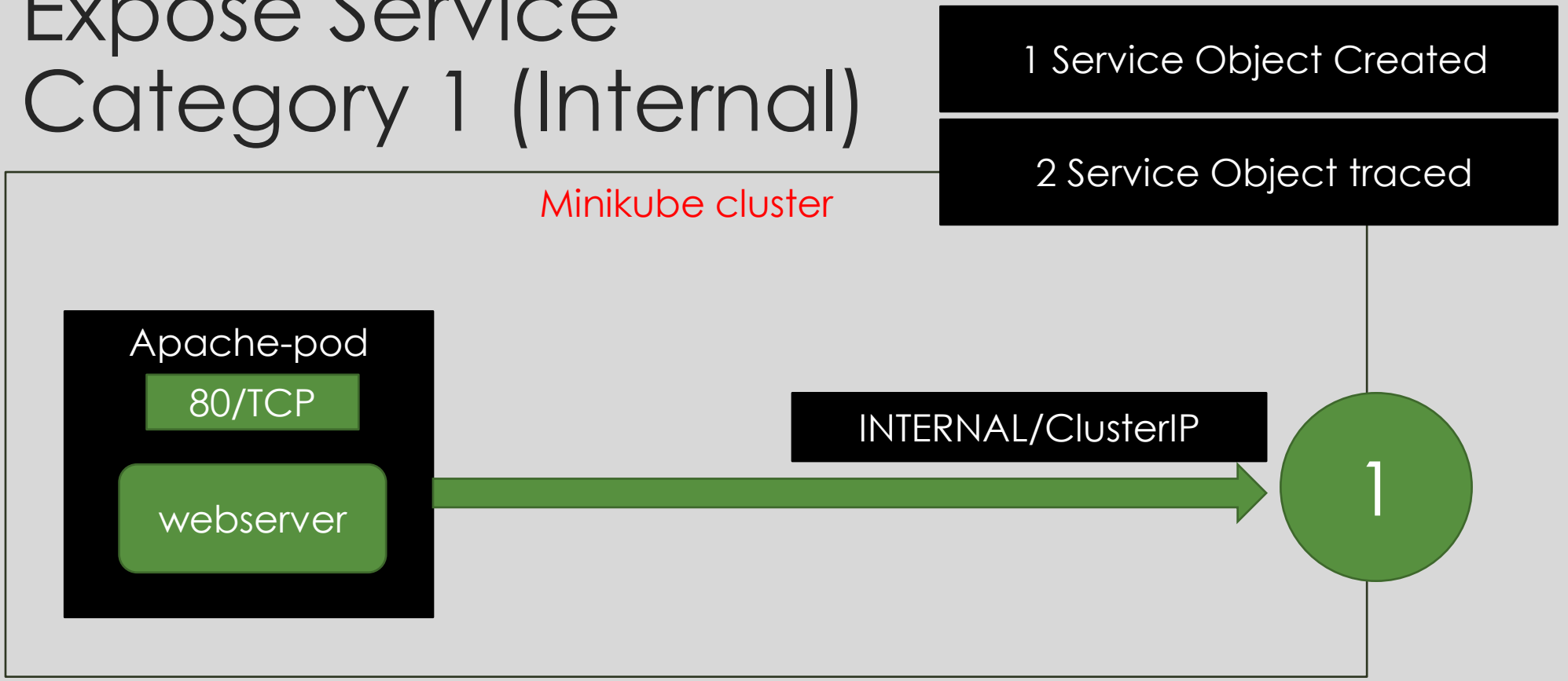
- Port Forward (Dynamic Port Fwd)→ 30000- 32767
- Service Name (Fully Qualified Service Name) – Service DNS name
  Service-Name.Namespace-name.svc.cluster.local
  /etc/resolv.conf (pod) → (Pod to Pod Communication)
- Service IP Address (Not recommended)
- External IP (Service) – Public cloud Gateway



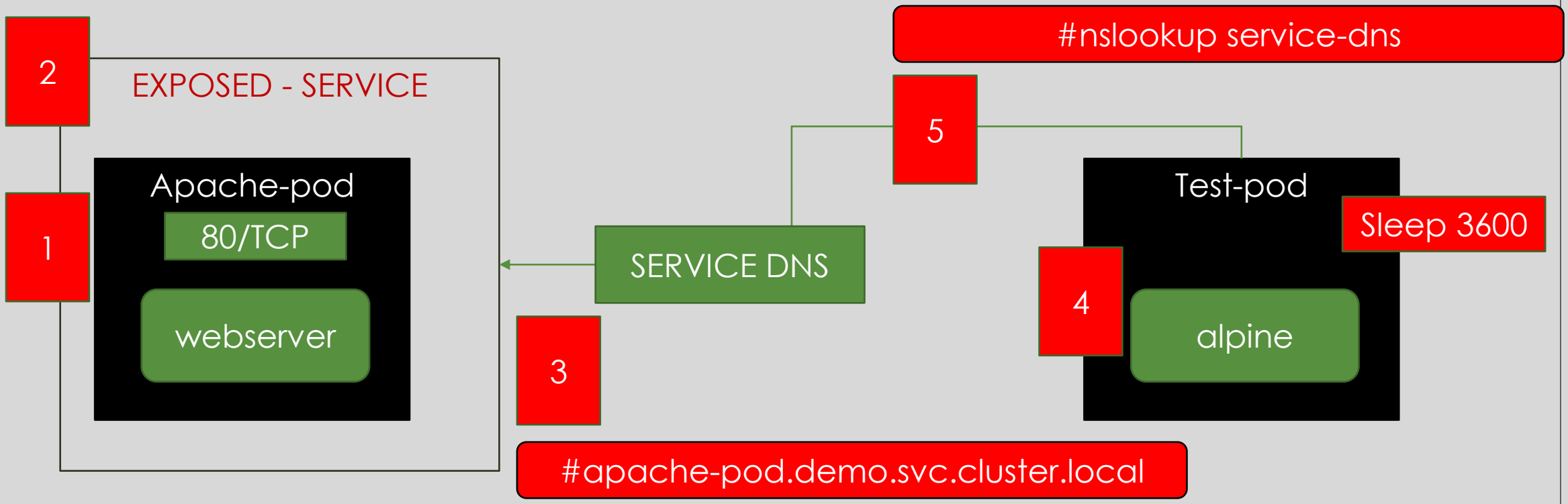
Expose Service Category 2 (External)



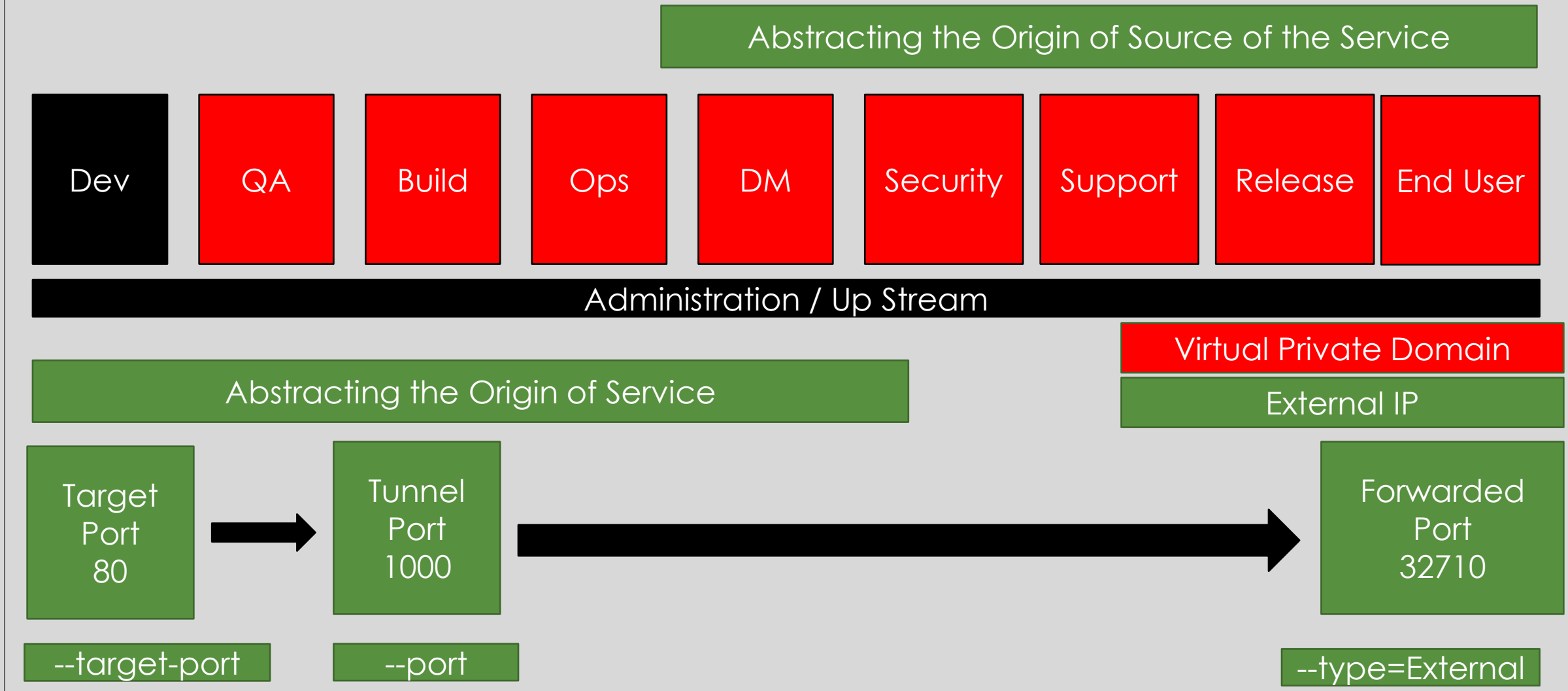
Expose Service Category 1 (Internal)



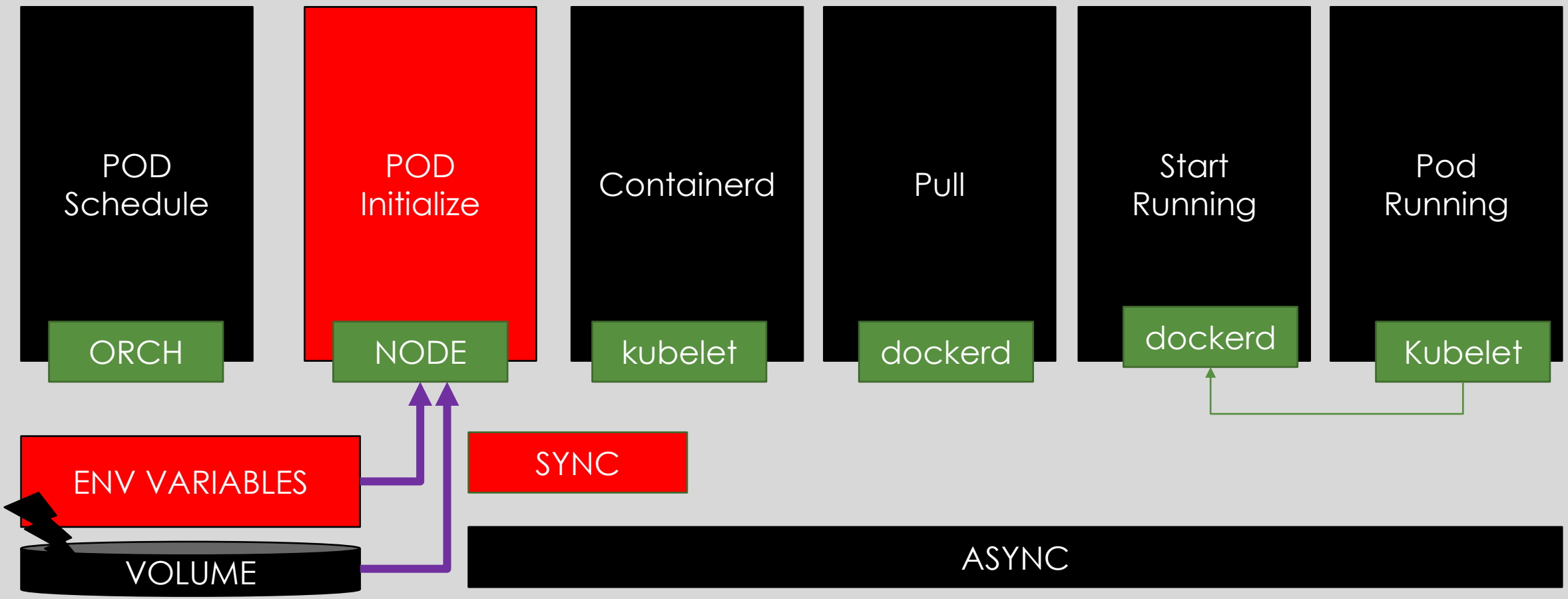
Pod to Pod Communication



Tunnel of Service

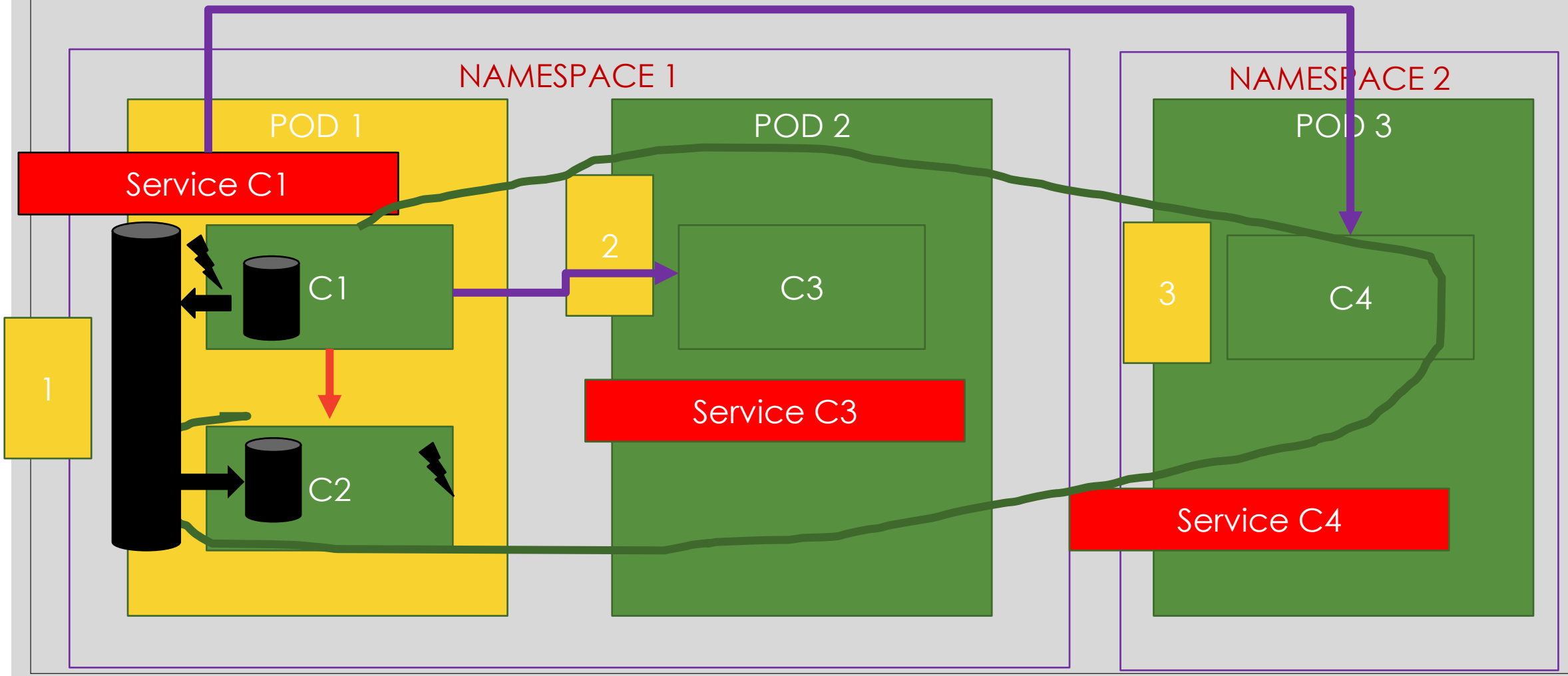


Lifecycle of Pod

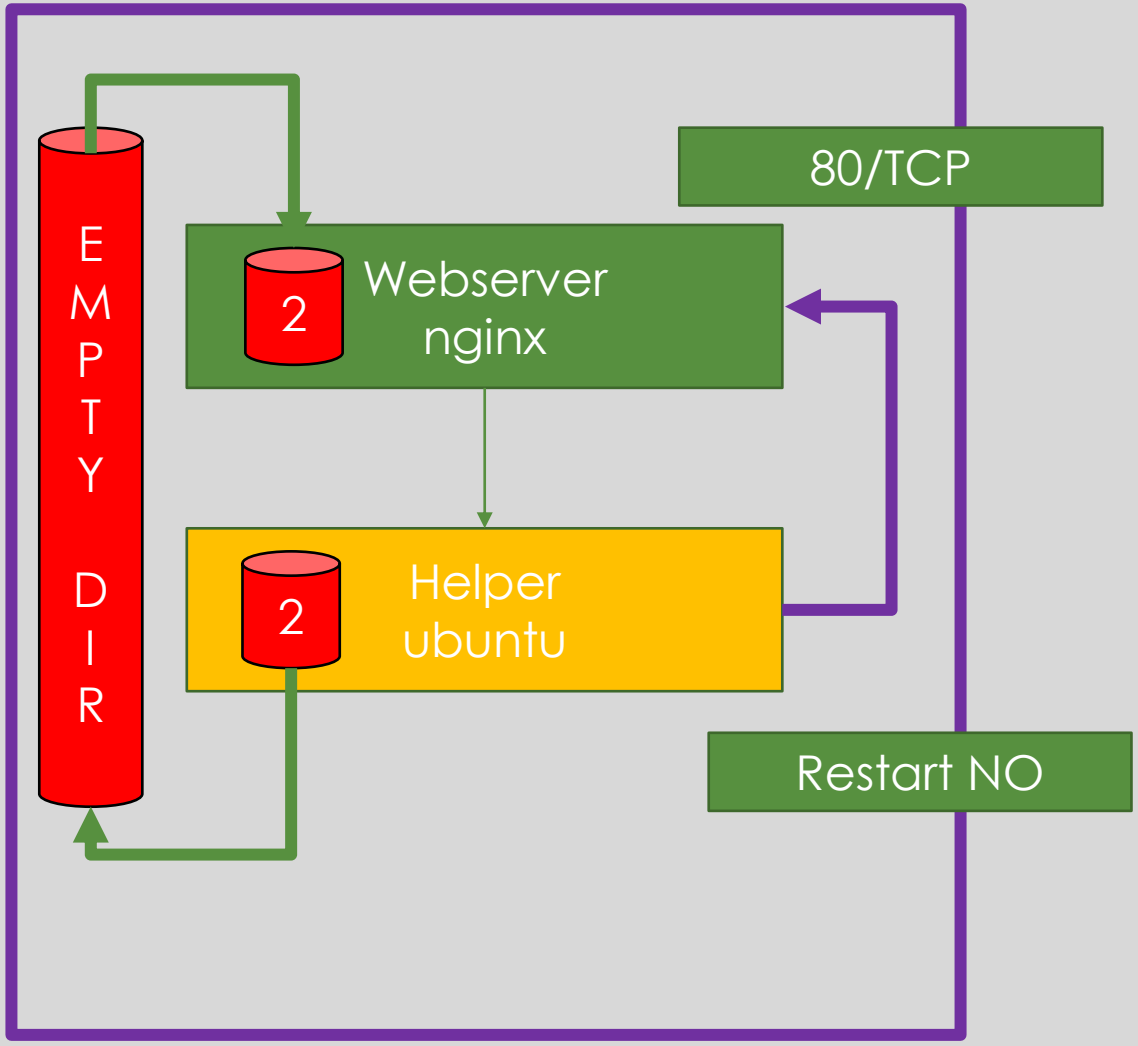


Communications

1 Communicating via VOLUME
2- SERVICE DNS ENTRY
3- SERVICE DNS ENTRY



Multi-container pod

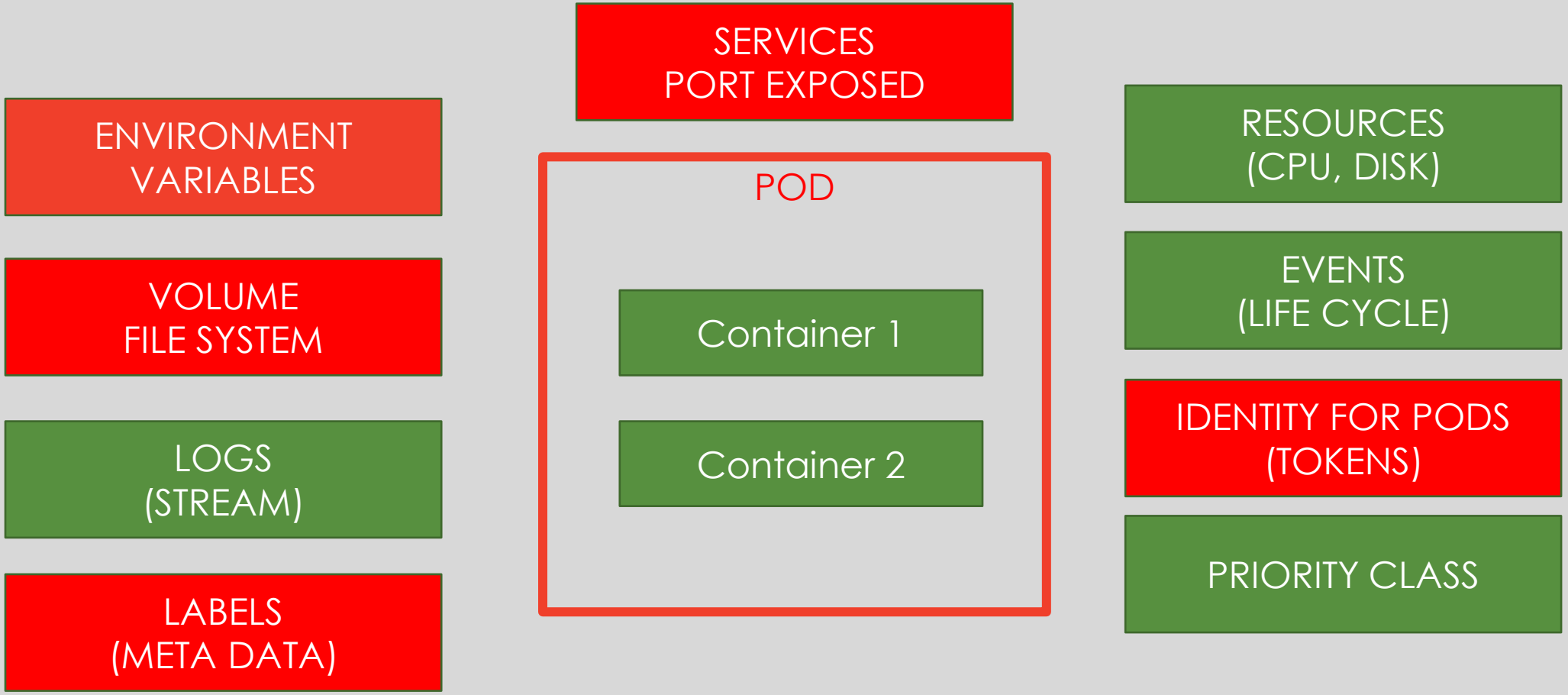


1 POD VOLUME – EMPTY DIR {}

2 CONTAINER HAVE MOUNTPOINT

3 RESTART POLICY NEVER

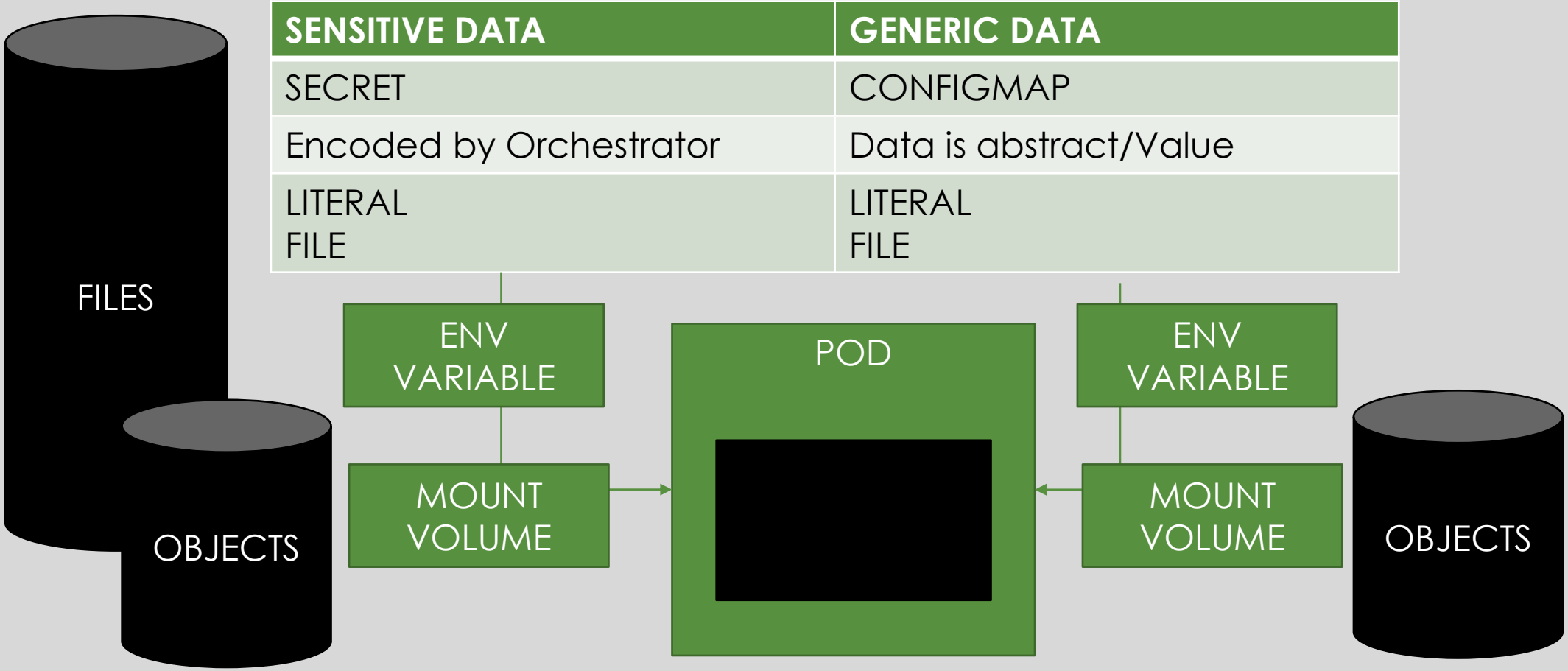
POD properties



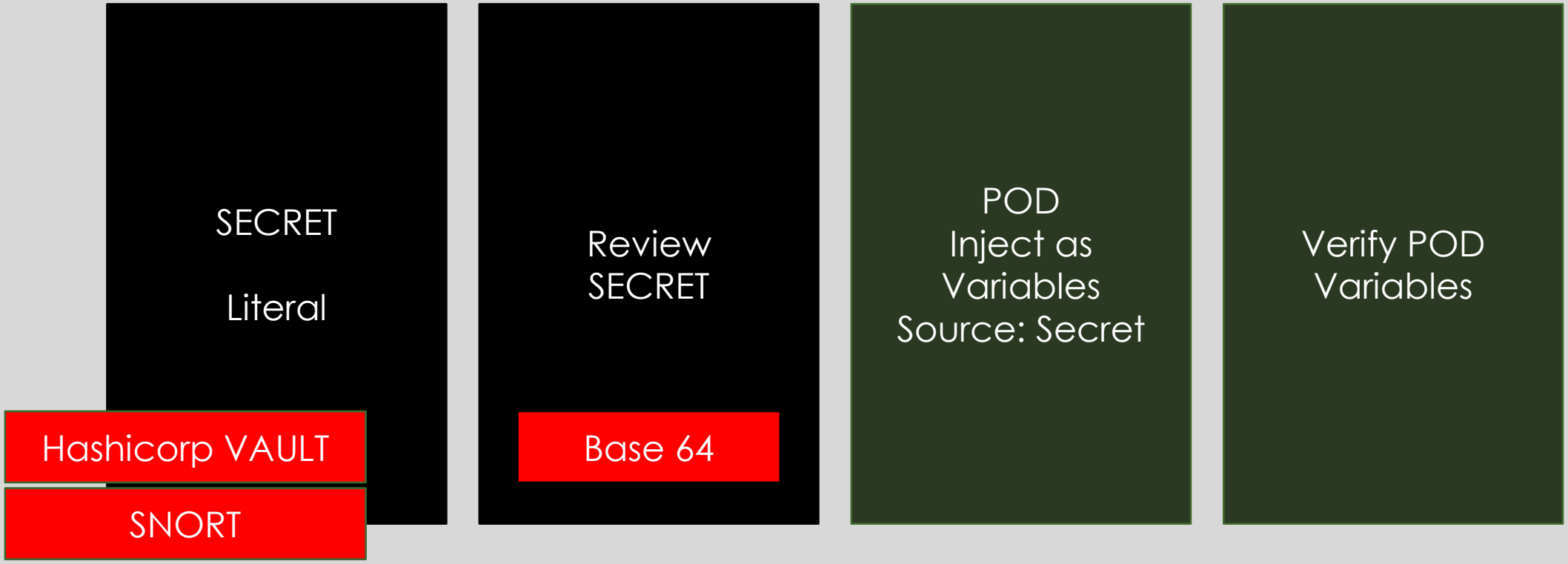
ENVIRONMENTS FOR PODS

WITHIN CONTEXT OF NAMESPACE

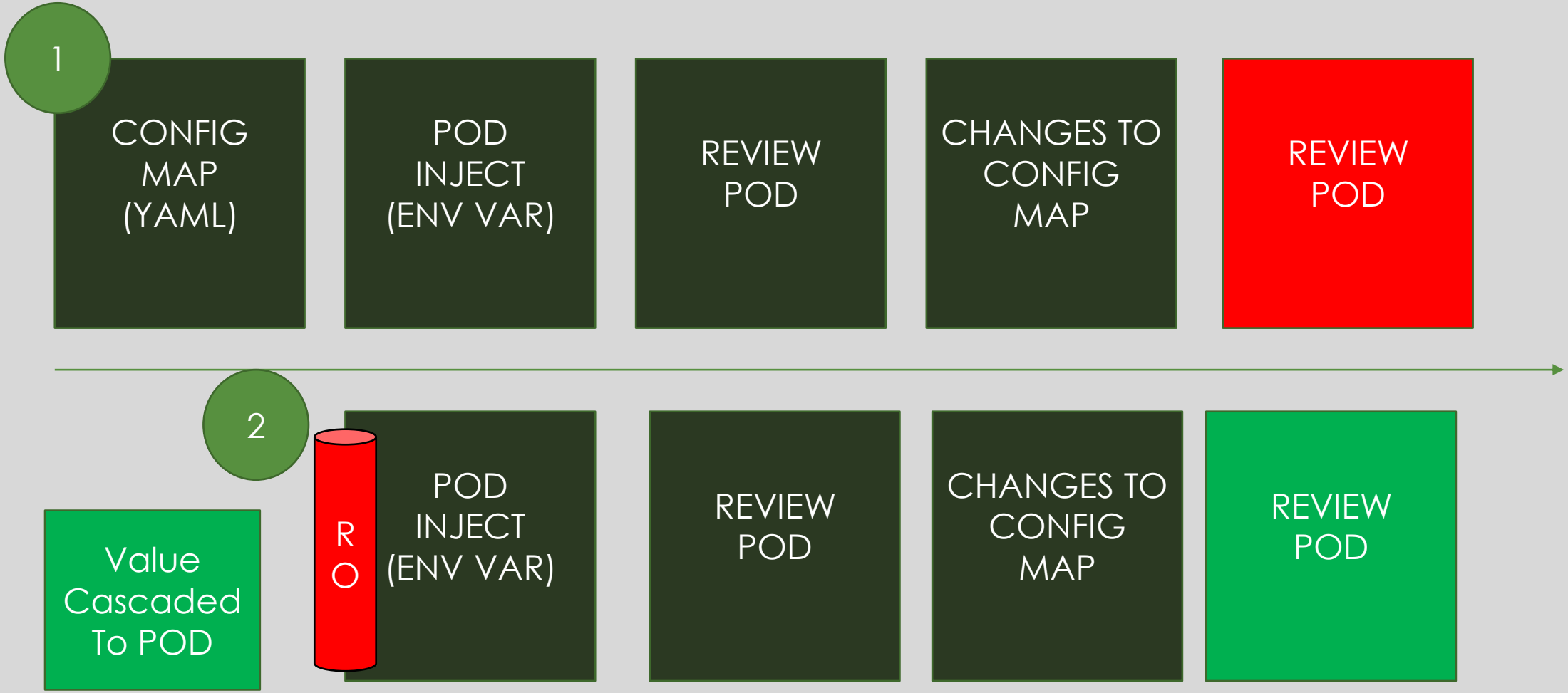
SENSITIVE DATA	GENERIC DATA
SECRET	CONFIGMAP
Encoded by Orchestrator	Data is abstract/Value
LITERAL FILE	LITERAL FILE



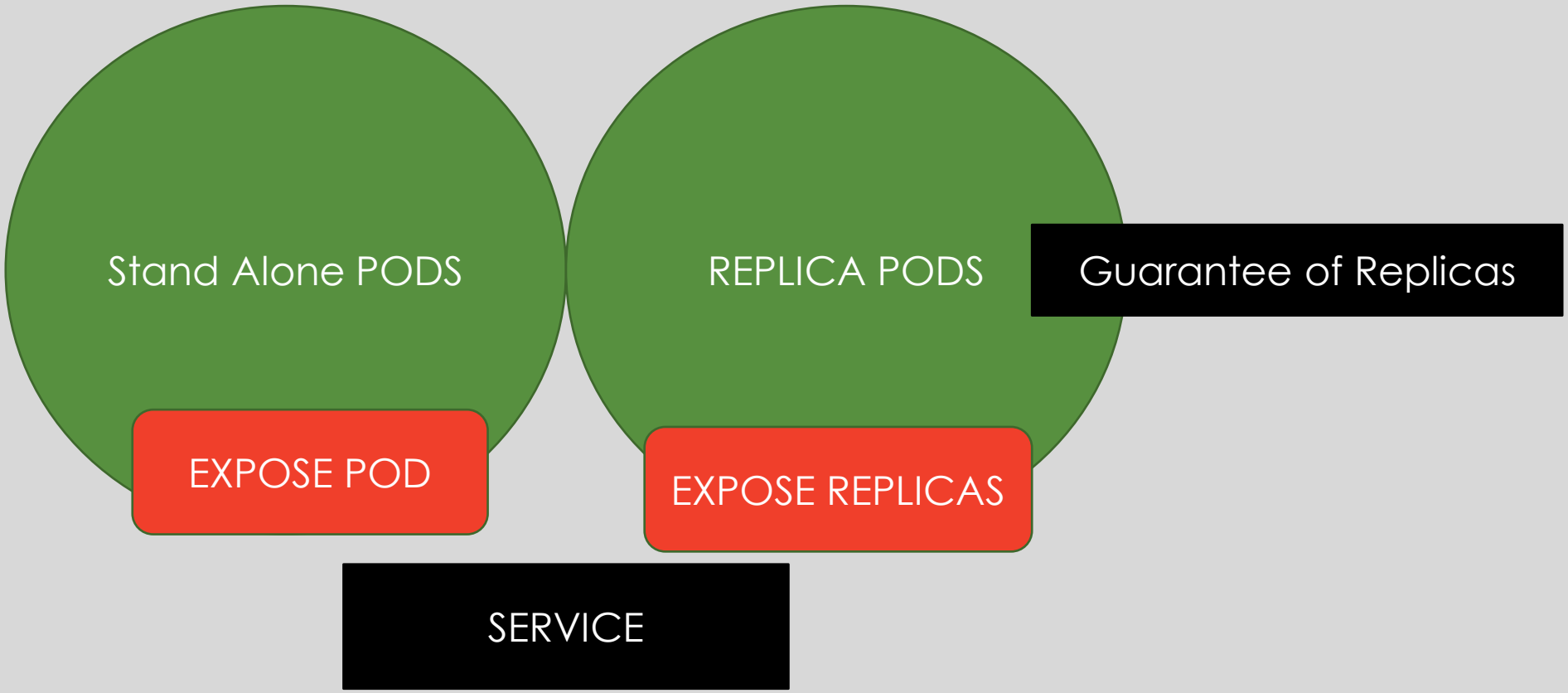
USE CASE ; SECRET injected TO POD



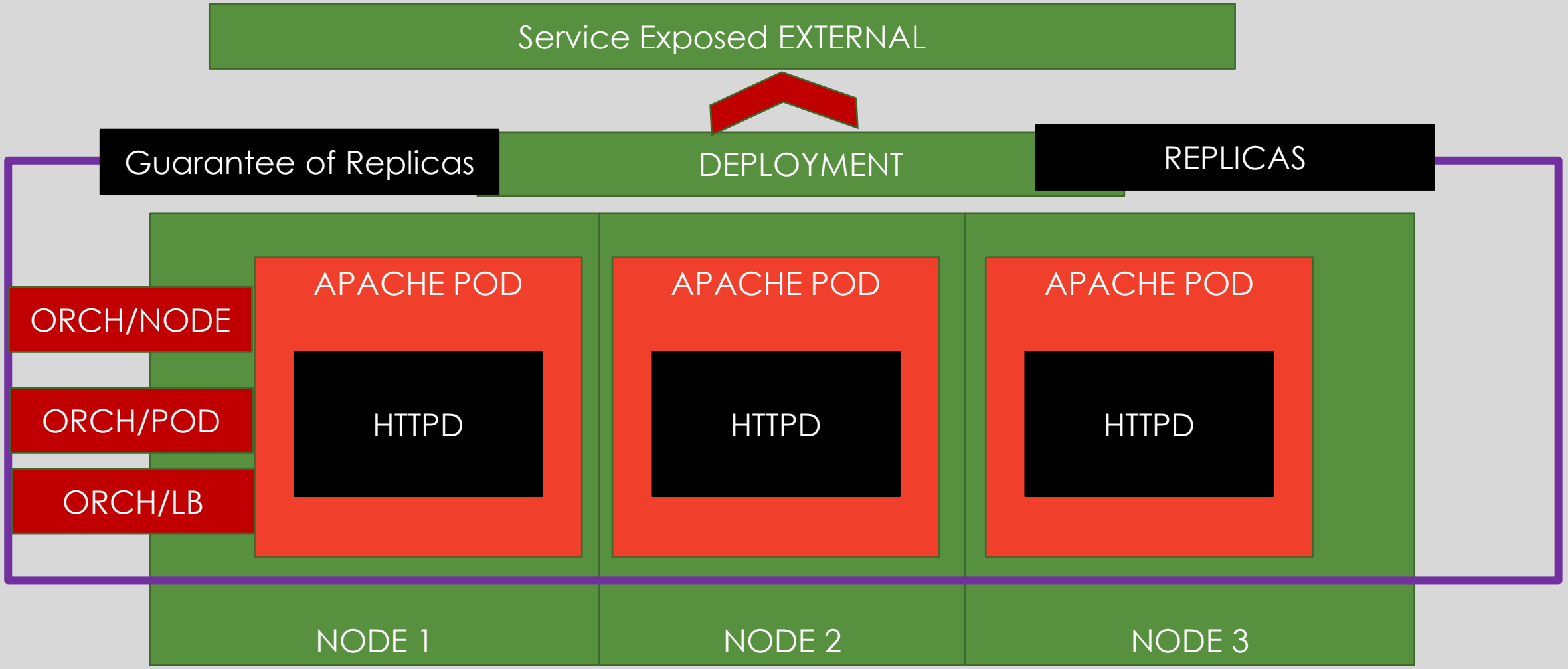
USE CASE : Config Map



Types of Pods...



Deployment


Service Exposed EXTERNAL
Guarantee of Replicas
DEPLOYMENT
REPLICAS
APACHE POD
APACHE POD
APACHE POD
ORCH/NODE
ORCH/POD
ORCH/LB
HTTPD
HTTPD
HTTPD
NODE 1
NODE 2
NODE 3

Troubleshooting in K8s

Before Object is created	During Object is Created	After Object is Created
Object Not created	Object created but not in desired state	Object in desired state , but outcome is not in desired state
1. Forbidden (RBAC) 2. Exceed Quota	ImagePullBack-Network down ErrImagePull- Invalid Image NotReady – one of container is terminated. VolumeMount – Volume Error	Docker Image ? CODE ?
#kubectl get events	#kubectl describe type name	#kubectl logs

THANK YOU