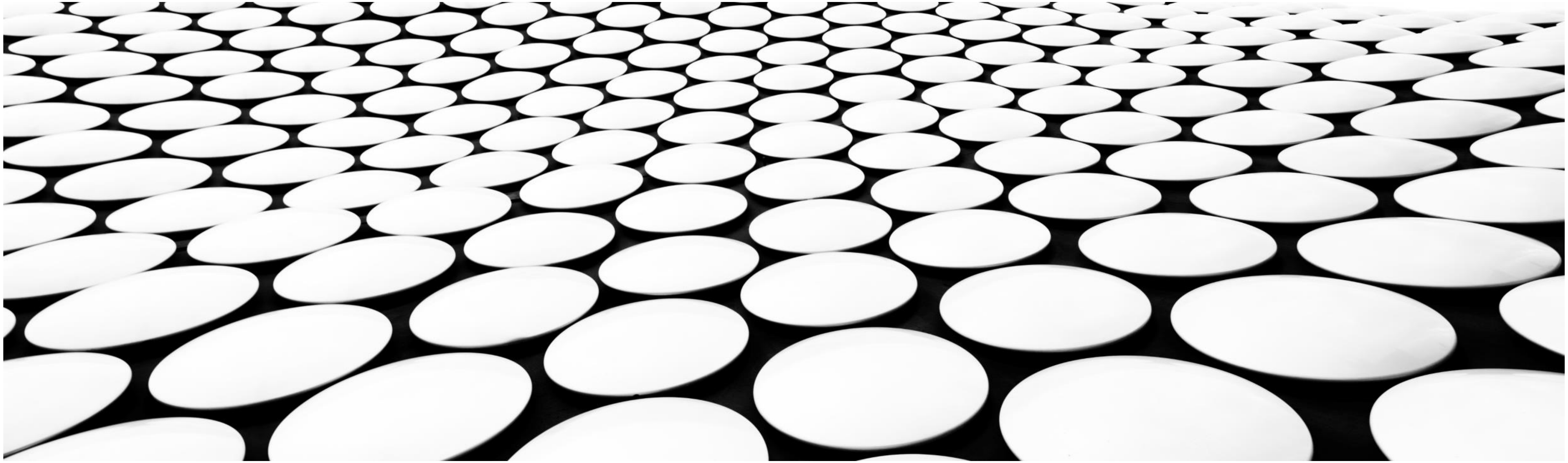
CONTAINER PIPELINES WITH ORACLE CLOUD

DJ (DHANANJAYAN)

DAY 3 – 01ST JULY 2020



DAY 3

- RECOVER IMAGES
- EXPORT IMAGES (SINGLE LAYERED)
- RESTART POLICY FOR CONTAINERS
- NETWORKING FOR CONTAINERS – CNI
- TROUBLESHOOTING FOR DOCKER
- KUBERNTES ARCHITECTURE
- INSTALL KUBERNETES
- NODE ARCHITECTURE

RESTART POLICY

NO (NEVER)	ALWAYS	ON-FAILURE
Container will not restart when implicit stop happens	Container will restart always when implicit stop happens	Service Fails – RESTART Retries Fixed
No Restarts	Indefinite restarts	Exit Code (exit 0 – Success) (code non-zero – Failure)

Explicit Stop will not restart containers

# docker stop

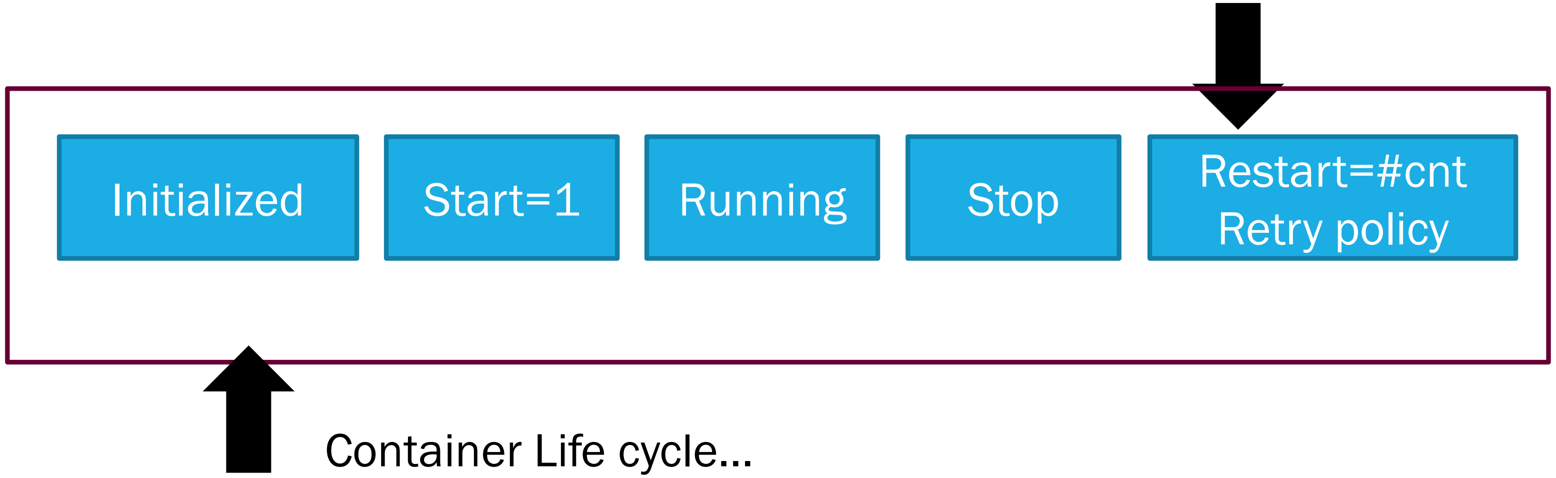
# docker kill

Implicit Stop ?

→ Memory Resources (Lack of)

→ Service abruptly stops

→ Attach → exit without detach keys



Container Life cycle…

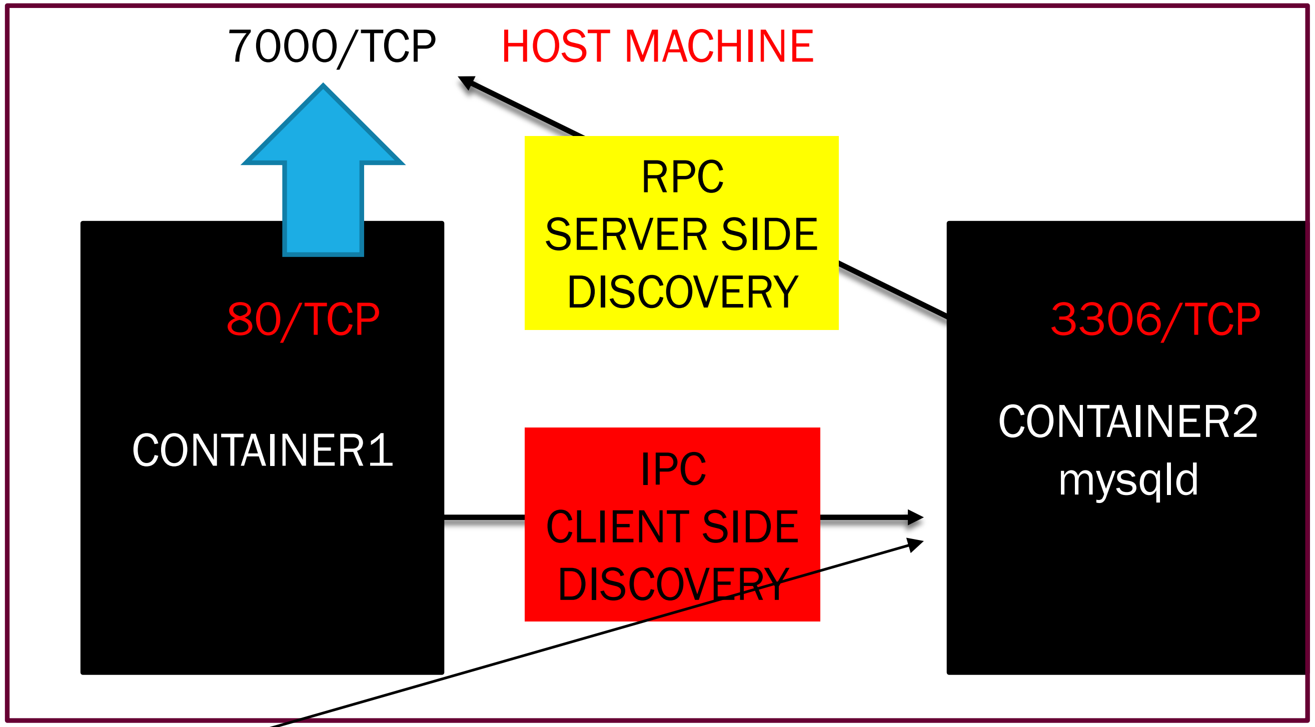
LOGS FOR CONTAINERS…

Image Logs → History of Image
Event Logs → dockerd logs (events)

Dev / QA	Administration	Support
Service UP ? Container Running? Container Parent Process Running?	Infrastructure Resources Scrutiny and Timestamp Who Executed ?	Diagnose Container Vs Image File System Change A Added C Changed D Deleted
# docker logs	# docker inspect LogPath	# docker diff <containername>
Parent process of container Environmental Variable change	Timestamp, user and application /sys/fs/cgroup/cpu/docker /sys/fs/cgroup/memory/docker Resource changes ?	File System change.

NETWORKING AS SERVICE

- COMMUNICATION AS SERVICE
- SUBNET OF **RUNNING C**ONTAINERS
- IP RANGE – IPV4 /IPV6 (CIDR)
- LOCAL
  - WITHIN MACHINE
  - **BRIDGE**, HOST (LINUX), NONE (DEPRECATED)
- VAN
  - BETWEEN MACHINES
  - OUTSIDE MACHINES
  - OVERLAY (CLUSTER)



7000/TCP

HOST MACHINE

RPC
SERVER SIDE
DISCOVERY

80/TCP

CONTAINER1

IPC
CLIENT SIDE
DISCOVERY

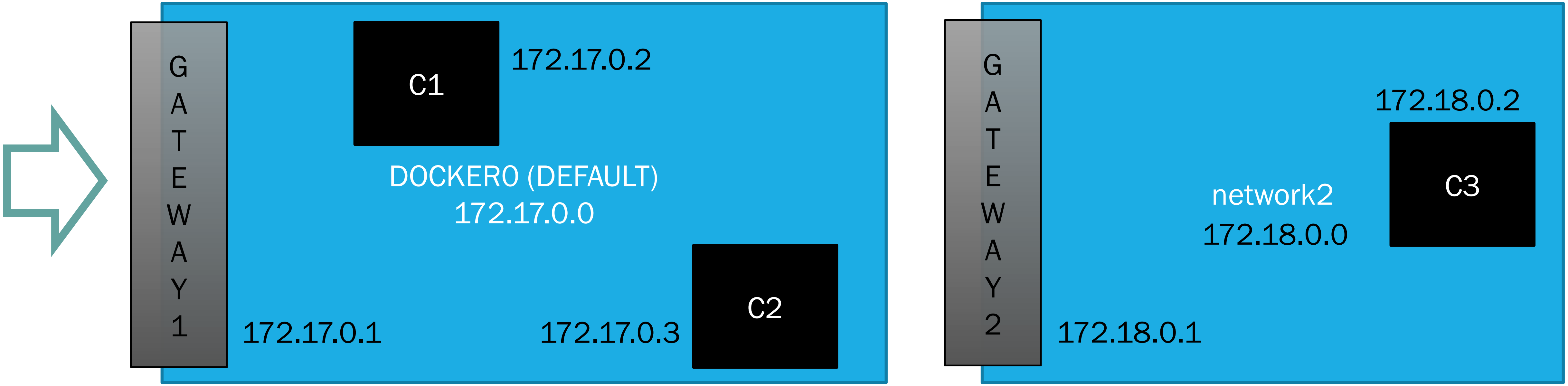
3306/TCP

CONTAINER2
mysqld

Container ID
Container Name
IPV4 ADDRESS
MAC ADDRESS (Router)
EndPoint Name (i-node)

172.X.Y.Z /16
256 x256 x 256 - 3
172.100.0.0 – 172.100.255.255
172.100.0.0 - Net Mask
172.100.255.255 – Net Mask
172.100.0.1 - Gateway

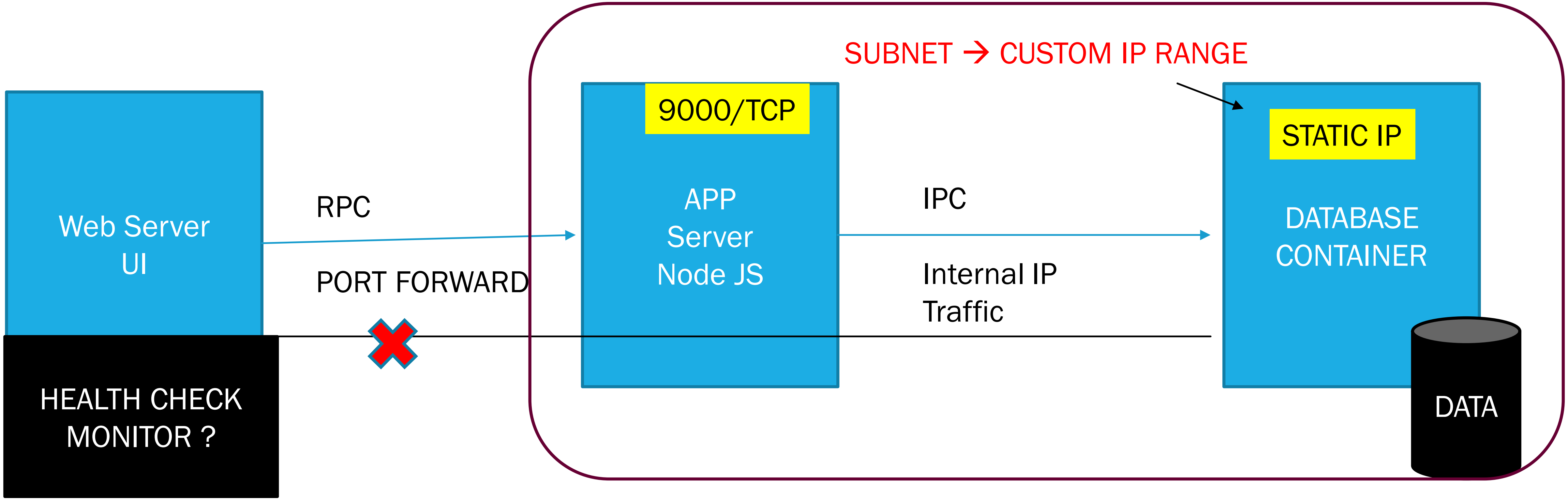
SCENARIO : 1



BOUNDARY OF RUNNING CONTAINERS LIMITED by SUBNET
GATEWAYS ARE PUBLIC
CONTAINERS WITHIN GATEWAY ARE PRIVATE

C1 → C2 (SAME SUBNET)
C1→ GATEWAY1 (PUBLIC)
C1 → GATEWAY 2 (PUBLIC)
C1 → C3 (DIFF SUBNET.. FAILURE)

USE CASE SCENARIO : 2



SUBNET → CUSTOM IP RANGE

IMPLEMENTATION

- Step 1 → Create a Network – CIDR Subnet - Custom Subnet – 172.100.0.0/16
- Step 2 → Build Docker Image for Database Container
- Step 3 → Create a Database Container with Static IP Address , Assigning to Network (Step 1) – 172.100.100.100
- Step 4 → Verify whether Database is running ?
- Step 5 → Seed Database IP in Node JS Code
- Step 6 → Build Docker Image for App Server
- Step 7 → Assign Node JS Container to Network (Step 1) , Port Forwarding
- Step 8 → Verify Service Output
- Step 9 → Create a Service Health Container.. With –health-cmd, --health-retries – health-interval –health-timeout=1s
- Step 10 → Verify Health of Service.

EXTRACT IMAGES

Data Center Backup	Release Management
Images with Layers Full Backup – Save and Load Retains with Layers as it is saved	Single Layer Architecture # docker import Compress all Layers into one layer
Backup / CI , CD	Release / CF

EASY SETUP…

Before	During	After
Setting up Application	Troubleshooting	Uninstalling
Shell Scripting (Batch) TerraForm (Script)	CLI - docker, docker-machine	TerraForm
YAML	CLI of docker	YAML

#docker-compose

Yaml – open sources, preferred by CNCF, approved by CNCF Providers
OCI , OEL, GIT, Platforms, Languages, Docker, Kubernetes, CNI, CSI , CM , DevSecops, Troubleshooting , CSP

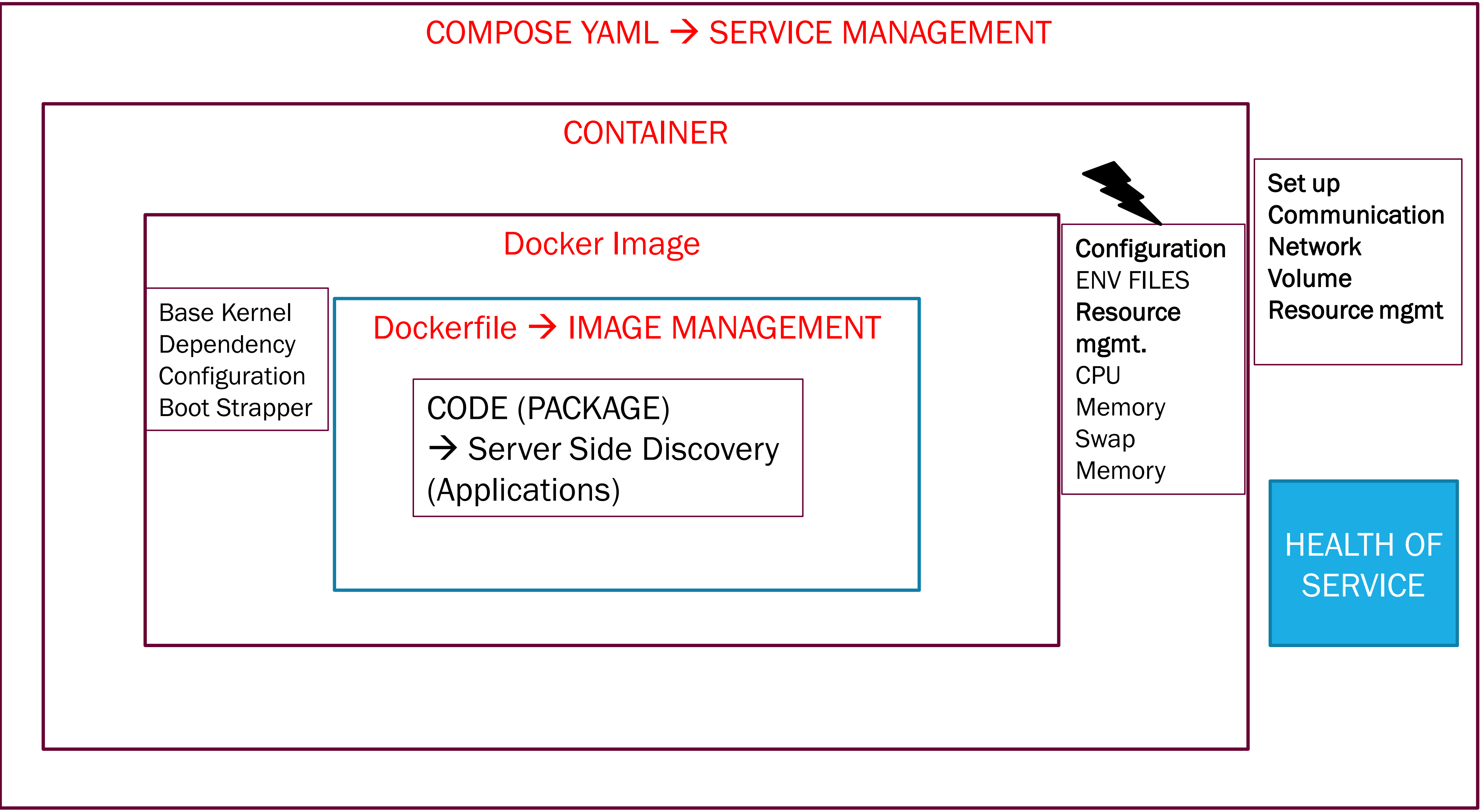
RULES IN YAML

- Indentation (Spacing)
- Case Sensitive
- Key: “Value”
- JSON :
- key: value → Scalar
- key : { rsa: xxx, pem: yyy } -→ “ | ”
- key : [collection] ➔ “-”
- Version : 3
- services

#docker-compose.yml

```
version: 3
services:
  database:
      image: newmysql
      environments:
      - MYSQL_ROOT_PASSWORD=admin
  web:
    image: httpd
    ports:
    - “8001:80”
    requests:
       memory: 200M
       cpus: 5
```

CONTAINER RUNTIME – MAKE UP FOR SERVICE



COMPOSE YAML → SERVICE MANAGEMENT

CONTAINER

Docker Image

Base Kernel
Dependency
Configuration
Boot Strapper

Dockerfile → IMAGE MANAGEMENT

CODE (PACKAGE)
→ Server Side Discovery
(Applications)

Configuration
ENV FILES
Resource mgmt.
CPU
Memory
Swap
Memory

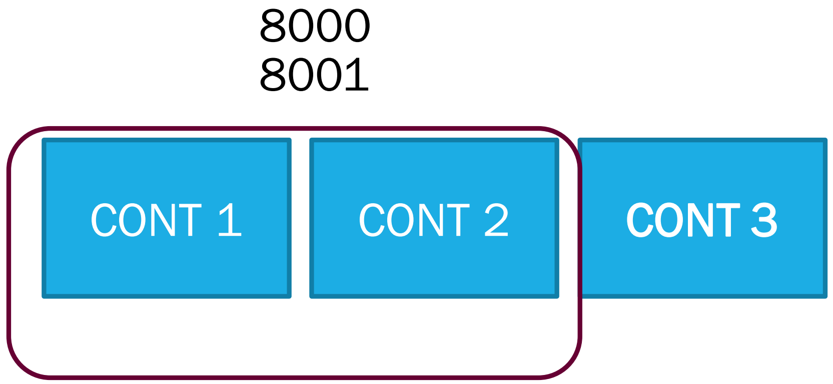
Set up
Communication
Network
Volume
Resource mgmt

HEALTH OF SERVICE

CONTEXT

DOCKER	K8s
MANAGE CONTAINERS – CONTAINERS ARE INDEPENDENT	MANAGE PODS – CONTAINERS ARE DEPENDENT ON PODS
CLI → Containers (docker CLI)	CLI → Manage (kubectl)
Restart Policy → NEVER (NO)	Restart Policy → ALWAYS , Self Heal (Recover, Repair)
Properties → JSON	Properties → Key Value Pair Format
Container Runtime → Docker	Any Container Runtime
Automation → YAML (Optional) , Manage CLI	CLI – 35-40% , Setup Applications (YAML)
No Scalability	Scalability - Horizontal or Vertical Scaling
Define Services as Docker Images/ Deploy as Containers	Define Services as Docker Images / Deploy them as PODS (which will internally contain containers)
Dev/QA (Development)	Operations/Administration

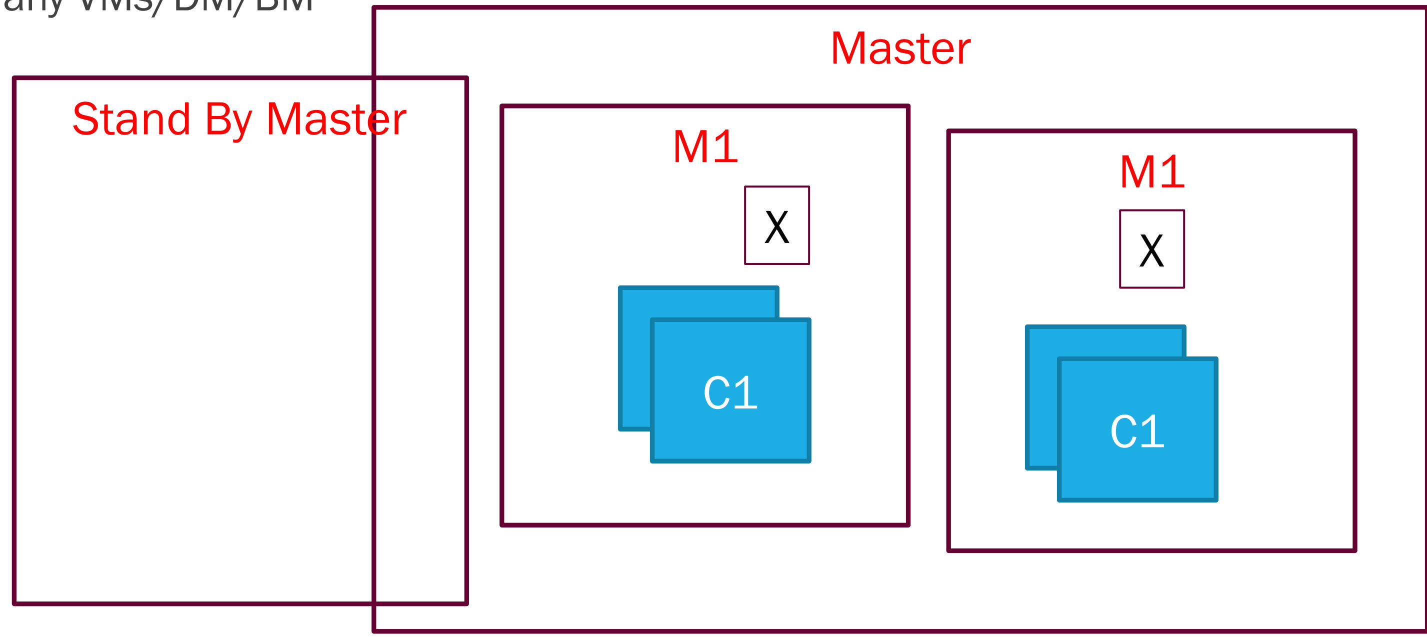
KUBERNETES



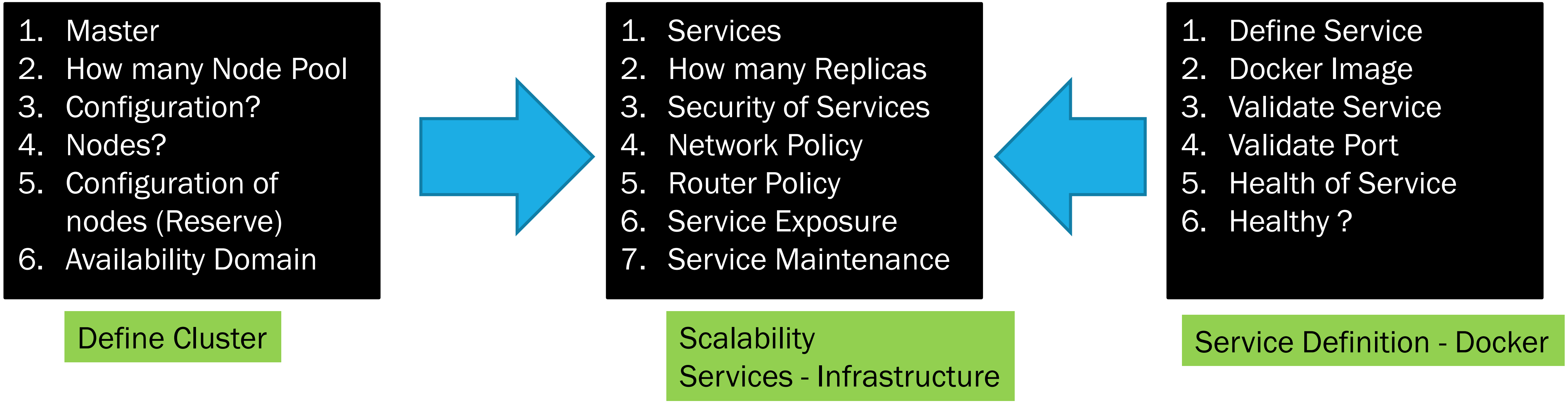
- Cluster of Container runtimes.
- Maintain High Availability
  - Services
  - Infrastructure
- Infrastructure – Replicate Infrastructure (SCALING OF VM/HOST)
- Replicate Services as Containers (Scaling of Containers) - End point of 8000 → Multiple Container Services
- Collection of Machine(s) – Networked together - Cluster
- Cluster is a collection of container runtimes .

CLUSTER

- Collection of Container runtime (dockerd) -- VM – Many VMs/DM/BM
- Leader of Cluster – Orchestrate
  - Monitor Utilization
  - Monitor Resources of Machines
  - Monitor Services
  - Monitor Health of Services
- Stand by Orchestrators
  - Leader Failure – Orchestator
- Machines (node)
  - Orchestrated Node
  - Worker Node



ROLES



1. Master
2. How many Node Pool
3. Configuration?
4. Nodes?
5. Configuration of nodes (Reserve)
6. Availability Domain

Define Cluster

1. Services
2. How many Replicas
3. Security of Services
4. Network Policy
5. Router Policy
6. Service Exposure
7. Service Maintenance

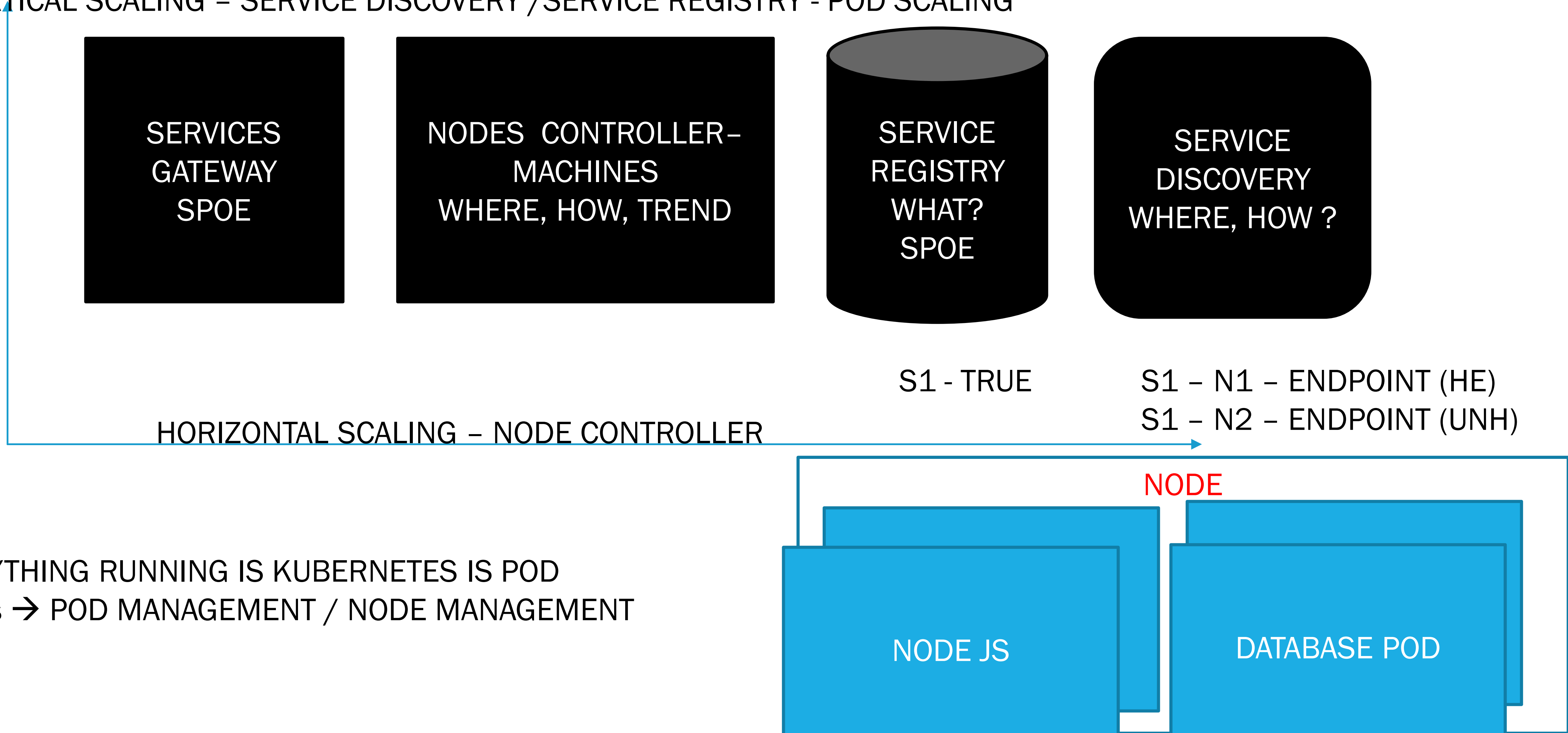
Scalability
Services - Infrastructure

1. Define Service
2. Docker Image
3. Validate Service
4. Validate Port
5. Health of Service
6. Healthy ?

Service Definition - Docker

ORCHESTRATOR NODE

VERTICAL SCALING – SERVICE DISCOVERY /SERVICE REGISTRY - POD SCALING



SERVICES
GATEWAY
SPOE

NODES CONTROLLER–
MACHINES
WHERE, HOW, TREND

SERVICE
REGISTRY
WHAT?
SPOE

SERVICE
DISCOVERY
WHERE, HOW ?

S1 - TRUE

S1 – N1 – ENDPOINT (HE)
S1 – N2 – ENDPOINT (UNH)

HORIZONTAL SCALING – NODE CONTROLLER

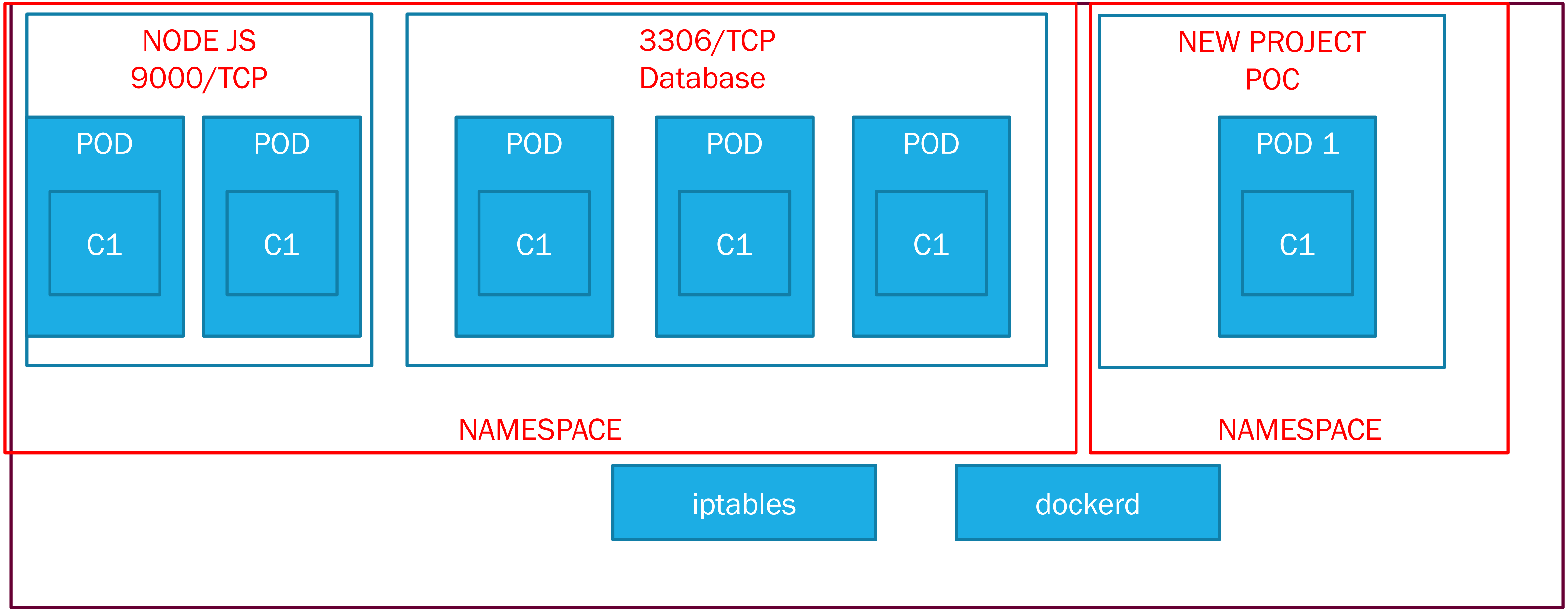
NODE

NODE JS

DATABASE POD

ANYTHING RUNNING IS KUBERNETES IS POD
K8s → POD MANAGEMENT / NODE MANAGEMENT

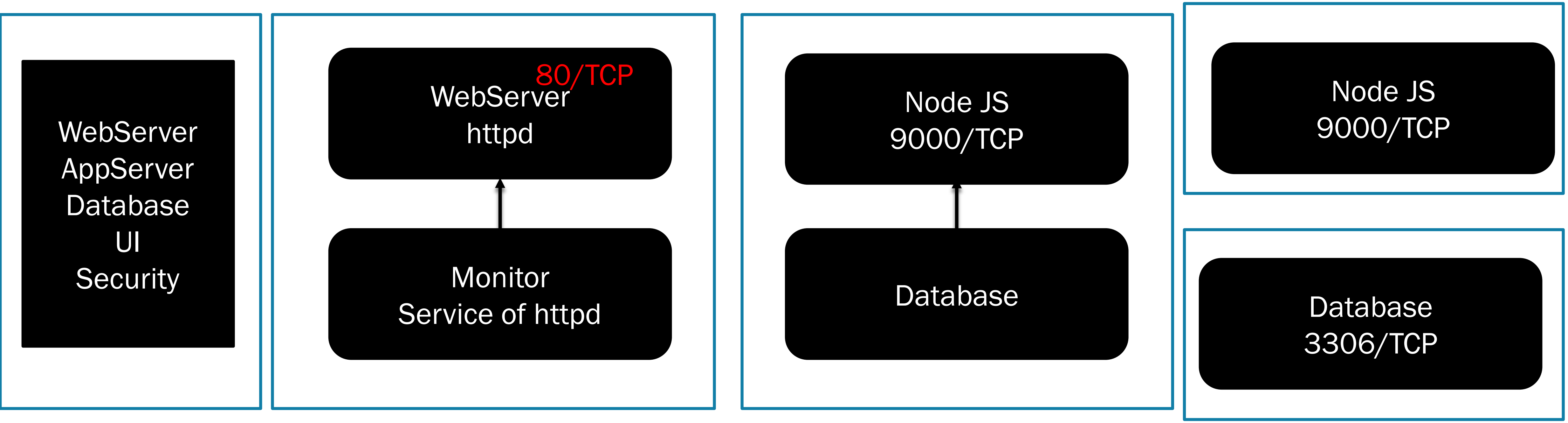
WORKER NODE


NODE JS
9000/TCP
POD
C1
POD
C1
3306/TCP
Database
POD
C1
POD
C1
POD
C1
NEW PROJECT
POC
POD 1
C1
NAMESPACE
NAMESPACE
iptables
dockerd

OBJECT OF ABSTRACTION

- MASTER
  - STAND BY MASTER
- NODE POOL
  - **NODES (VM)**
    - NAMESPACES
      - PODS
        - CONTAINERS
          - **IMAGE (CODE)**
            - LAYERS

UNIT OF SCALE (ABSTRACTION) - POD



WebServer
AppServer
Database
UI
Security

80/TCP
WebServer
httpd

Monitor
Service of httpd

Node JS
9000/TCP

Database

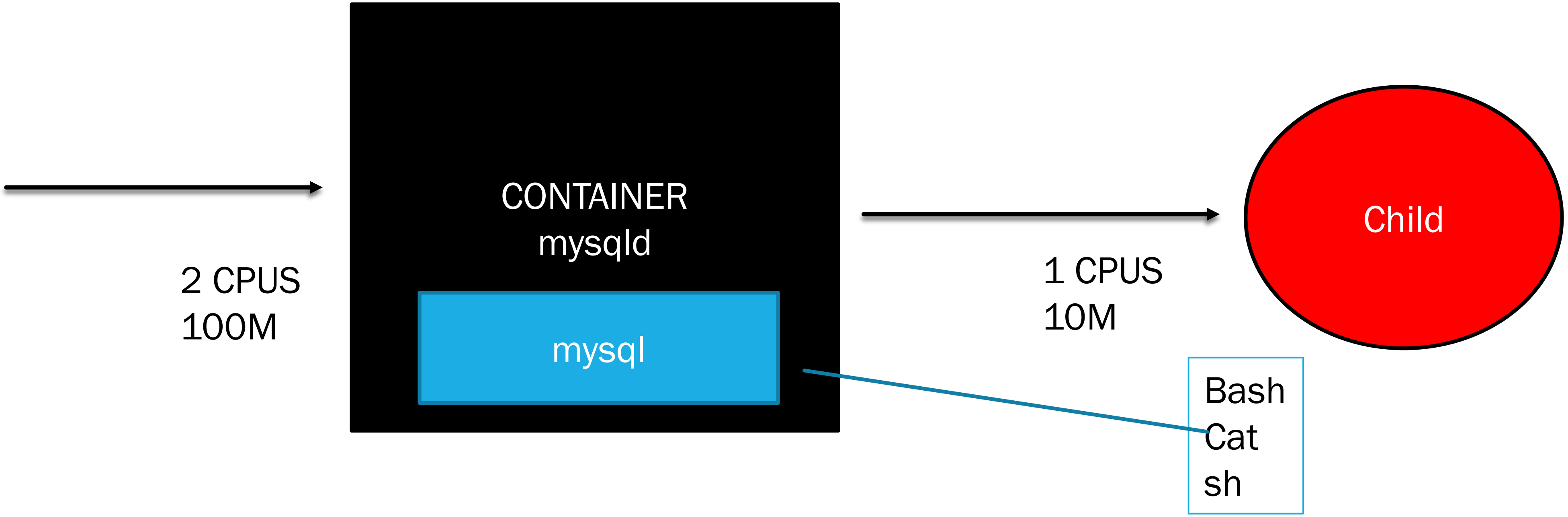
Node JS
9000/TCP

Database
3306/TCP

SCALE A PORT (SERVICE) → MANY CONTAINERS
SCALE A SERVICE → ONE CONTAINER
SCALE A SERVICE → MANY PORTS – ONE OR MORE CONTAINERS

Java – Jar → Code -→>>. WebLogic (Jar) → Database → Cloud (on Prem)
Docker images (Jar) → JRE , Database (Cache), WebLogic (Cloud)

TESTING ON CONTAINER.. EXEC CHILD PROCESS



2 CPUS
100M

CONTAINER
mysqld

mysql

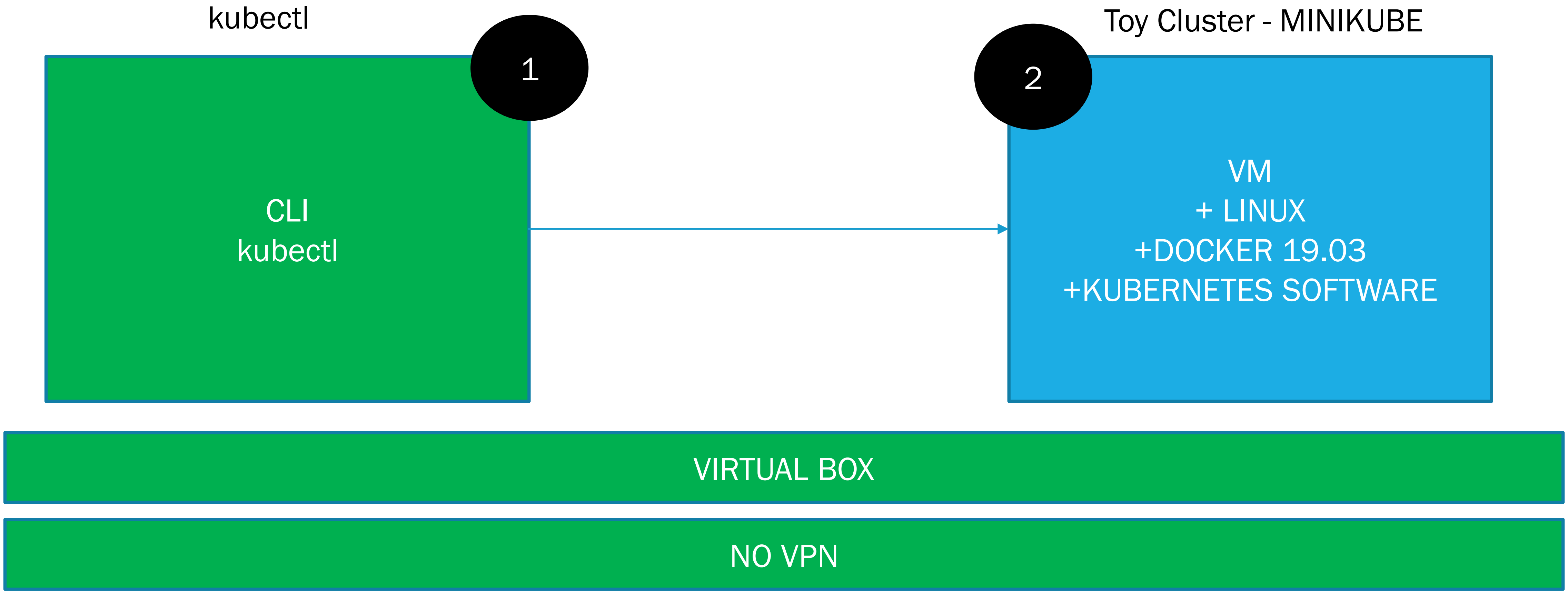
1 CPUS
10M

Child

Bash
Cat
sh

SIMPLE STEPS

```
#minikube start –cpus=2 –memory=3072 –vm-driver=virtualbox
```



kubectl

1

CLI
kubectl

Toy Cluster - MINIKUBE

2

VM
+ LINUX
+DOCKER 19.03
+KUBERNETES SOFTWARE

VIRTUAL BOX

NO VPN