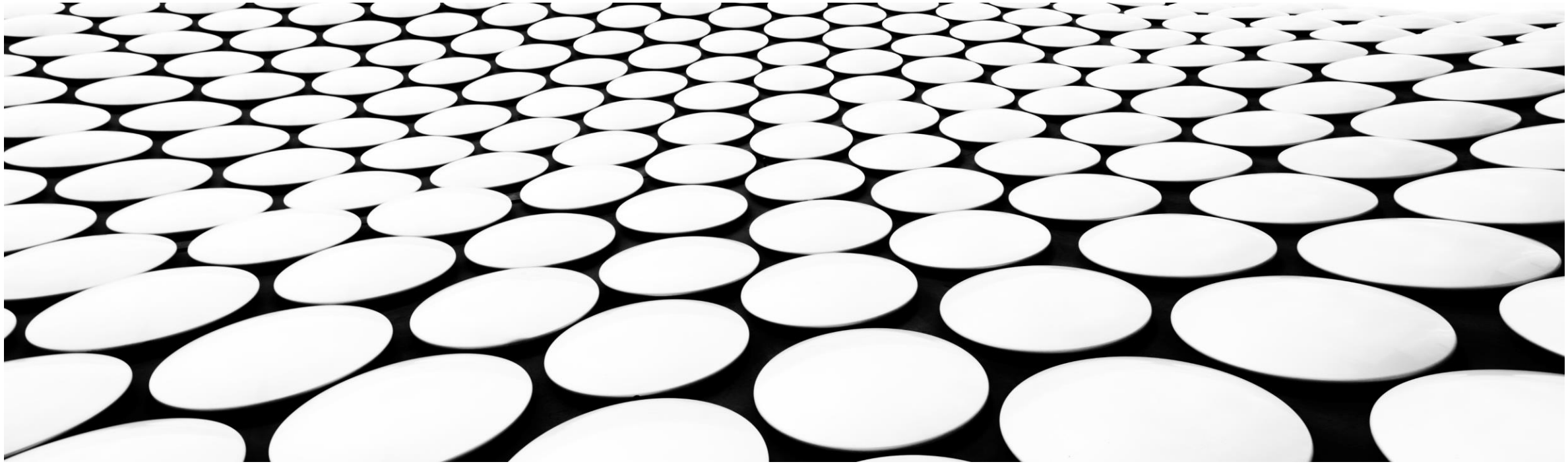
DOCKER AND KUBERNETES

DHANANJAYAN

20TH JULY – 24TH JULY 2020



DAY 2

- Custom Image Builds
- Automation
- Troubleshooting
- Database Image (Setup)
- App Server (Setup) – MSA
- Sharing Images
- Port Forward as Service
- Databases and Microservices

AUTOMATION

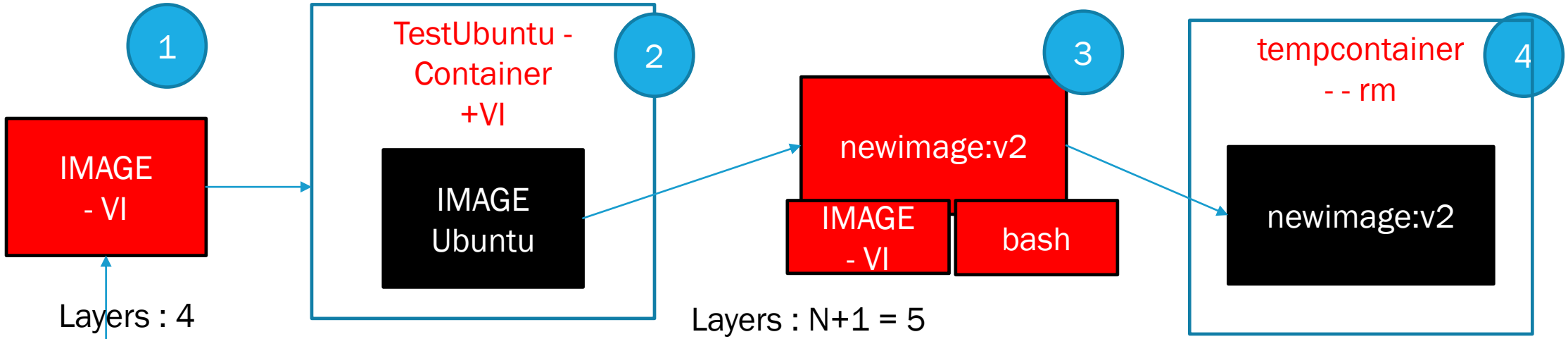
- `expression`
- $(expression)
  - # docker start $(docker ps –a –q)
- {{Extraction from JSON}}
  - {{.Parent_Property.ChildProperty}}
  - -f (Formatted JSON)
  - docker inspect -f "{{.Name}} {{.State.Status}} {{.NetworkSettings.IPAddress}}" \

$(docker ps -a -q)

LOGS

Status of the Container	Administrative View	Support View	Event Logs
Container Stats – Inside the Container			dockerd
Container Running ?	Scrutiny – User , Timestamp, which	Diagnose	Events of docker – 7 Docker Object status with respect to dockerd
Logs of **Parent Process** – Output of Parent Process	Parent Process + more details	Audit on File System A – Added C- Changed (Permission) D – Deleted	Trace dockerd
JSON-FILE	JSON-FILE	File system changes	--since
Dev. QA	Admin, Operations and Release	Release and Support	Docker administrator
# docker logs	# docker inspect LogPath	# docker diff	# docker events

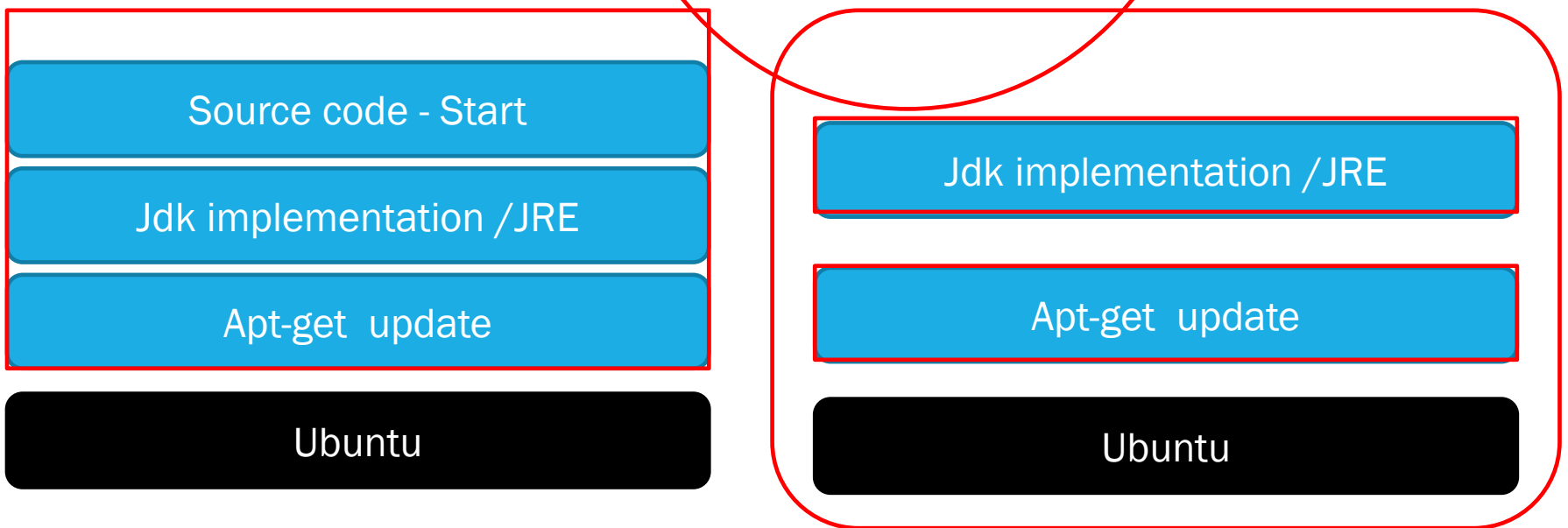
CUSTOM IMAGE BUILD - SOLUTION 1



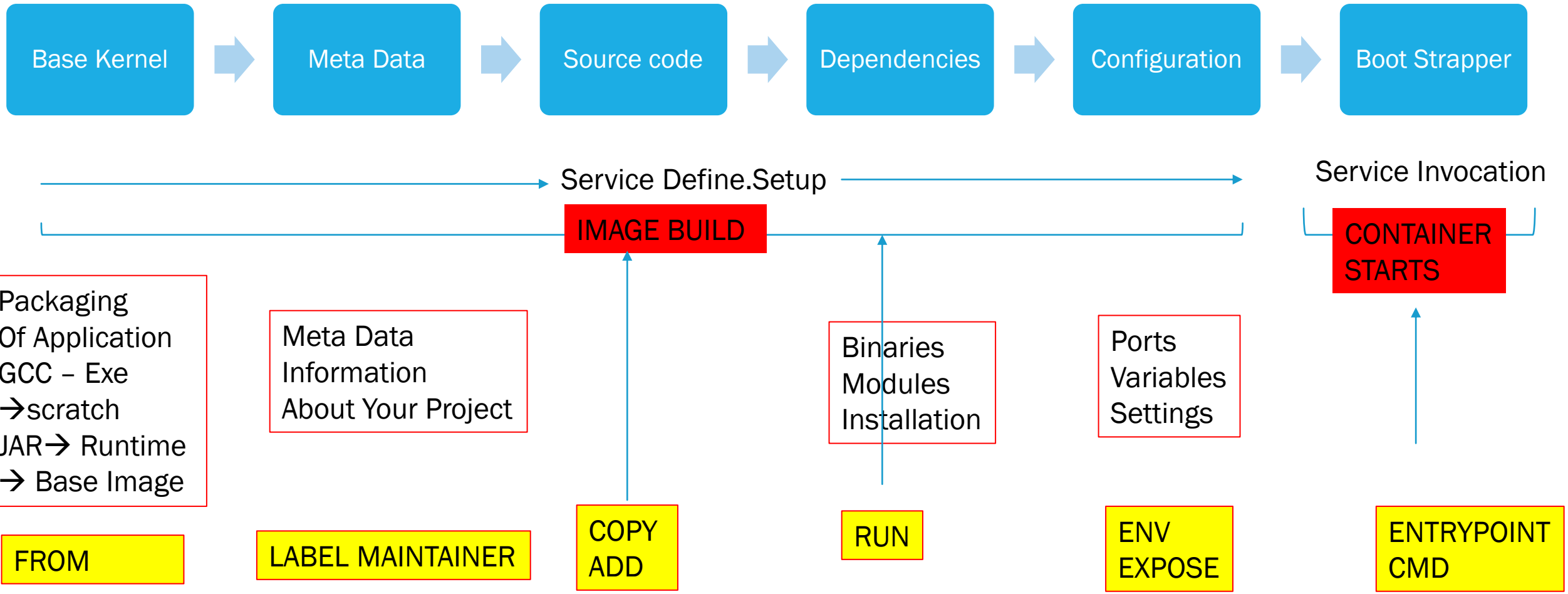
Number of Layers : + 1 when committed
Synchronized – run as Foregound
Not Standardized for Automation
Manual pruning of images

CUSTOM IMAGES

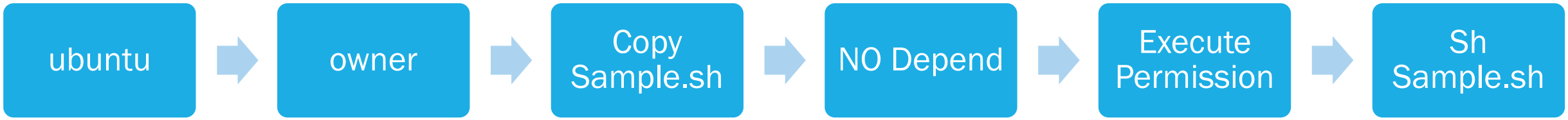
DOCKER COMMIT	Dockerfile (SOP)	VM
Container to Docker Image +1 Layer Foreground Process (Slow) API Support ? +Standardizing ?	+ API Support +Choice of Layers + Background +Automation + CNCF	VM → Docker Image Rancher VM→ Root FS → Docker Image (Monolothic) COST ($)



DOCKER IMAGE (DOCKERFILE CONSTITUENTS)



USE CASE : SHELL SCRIPT (IMPLEMENTATION)



Dockerfile Keywords – Case Neutral
Dockerfile → Dockerfile (Standard)
Dockerfile → Comments → #

Execute :
Dependencies → are Loaded into Build Cache of dockerd → Background → CONTEXT_ROOT (Path of the dependencies)

docker build –t <<NameofImage_Lowercase:tag_name>> -f <<Dockerfilename>> [[CONTEXT_ROOT_PATH]]

MITIGATIONS (USE CASE 1)

Dockerfile Keywords – Case Neutral
Dockerfile → Dockerfile (Standard)
Dockerfile → Comments → #

Execute :
Dependencies → are Loaded into Build Cache of dockerd → Background → CONTEXT_ROOT (Path of the dependencies)

docker build –t <<NameofImage_Lowercase:tag_name>> -f <<Dockerfilename>> [[CONTEXT_ROOT_PATH]]

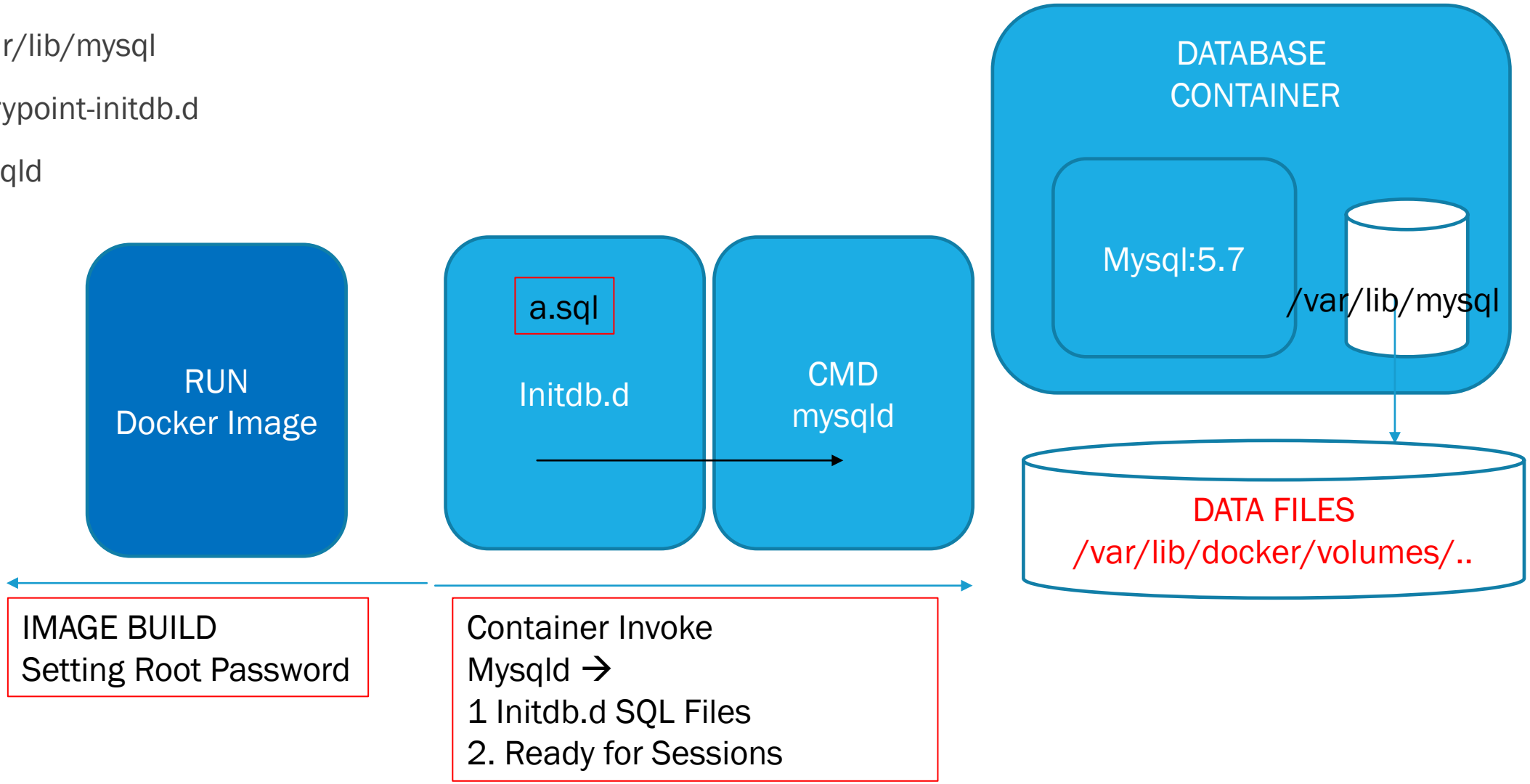
RISKS:
1. Disclosing Source Code
2. Enters into Shell of Container.

Sh /code/Sample.sh /etc/hosts

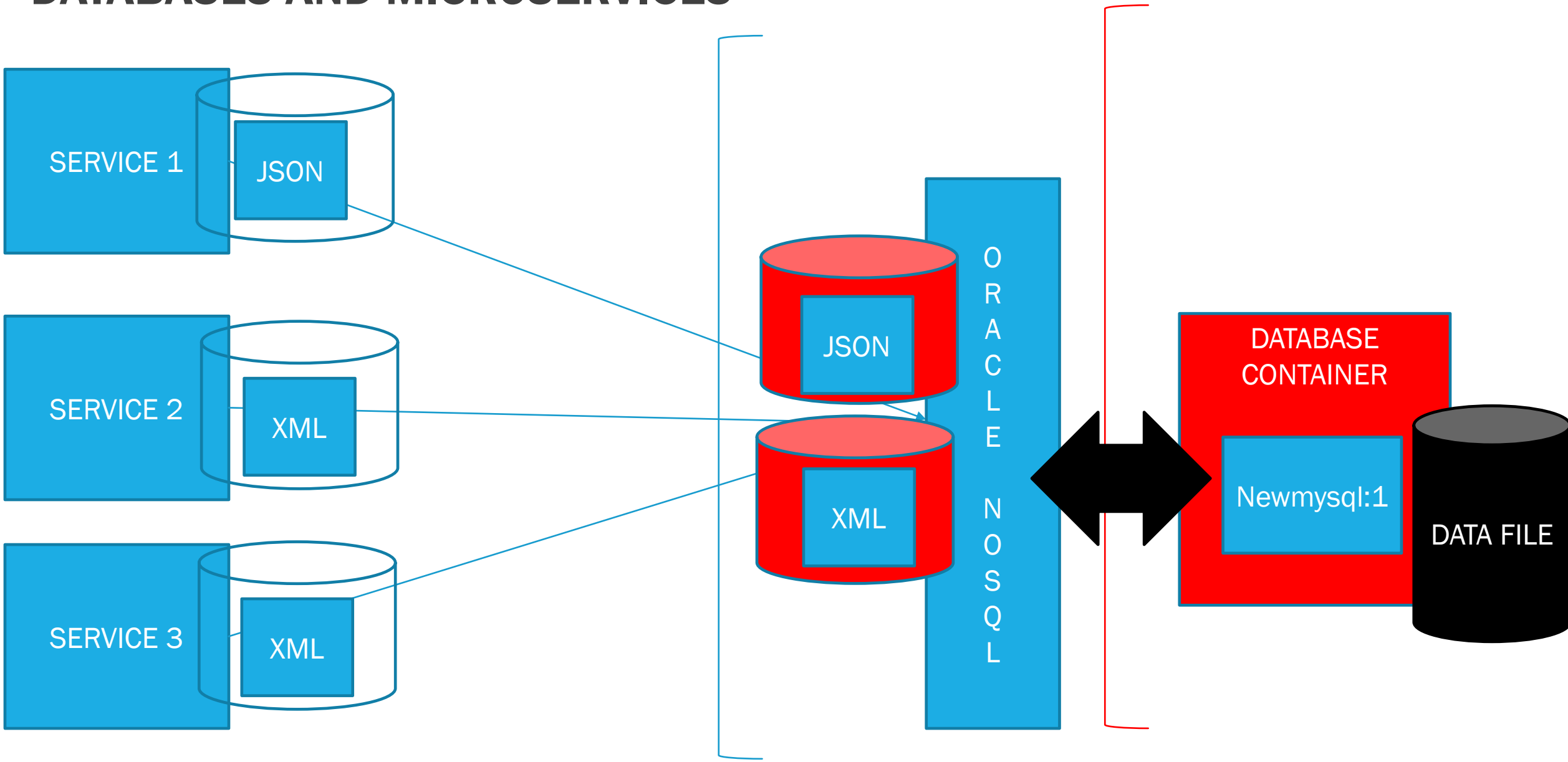
CMD → Variable Component of Boot Strapper
ENTRYPOINT → Fixed Component of Boot Strapper
[Array /List of Strings]
ENTRYPOINT [“sh”,”/code/Sample.sh”]
CMD [“/etc/hosts”]

USE CASE: DATABASE

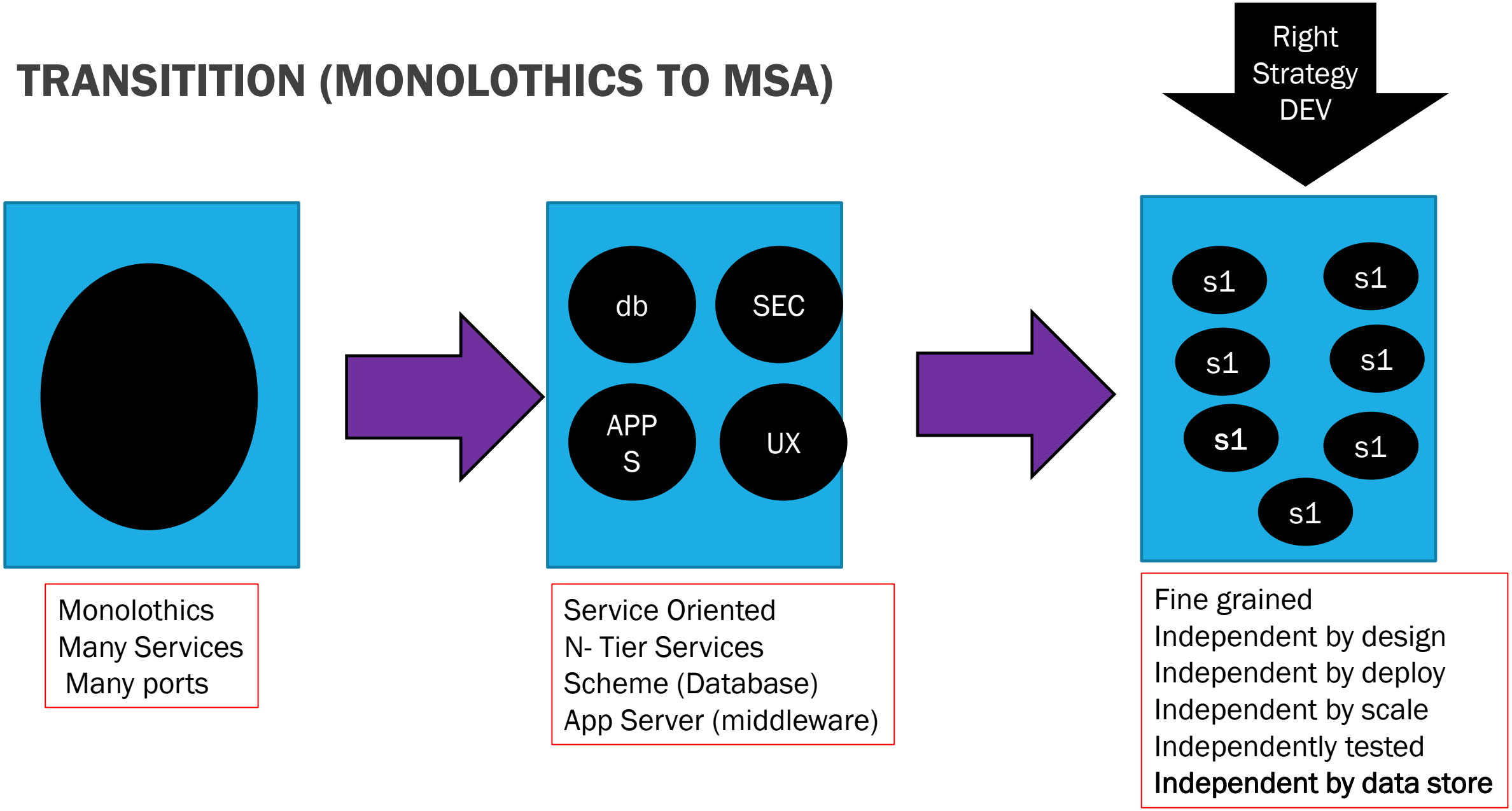
- ENV MYSQL_ROOT_PASSWORD
- VOLUME /var/lib/mysql
- /docker-entrypoint-initdb.d
- CMD → mysqld



DATABASES AND MICROSERVICES



TRANSITITION (MONOLOTHICS TO MSA)



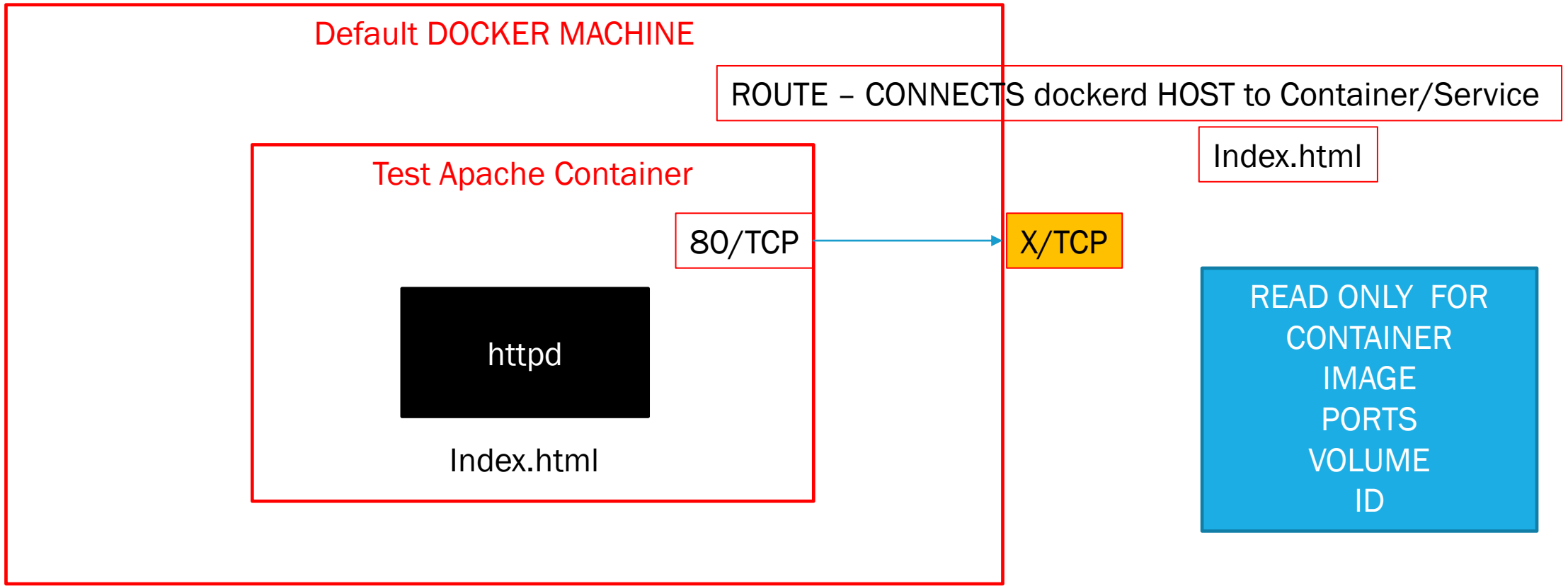
Monolothics
Many Services
Many ports

Service Oriented
N- Tier Services
Scheme (Database)
App Server (middleware)

Fine grained
Independent by design
Independent by deploy
Independent by scale
Independently tested
Independent by data store

PORT FORWARDING...

SERVICE CALLS – 1	SERVICES – CALL – 2
SYNC / REST CALLS .1-1 timeout 100,200 , 300,400,500,600*	ASYNCHRONUS / MQ 0- # /timeout.. ? (Publisher – Subscriber)



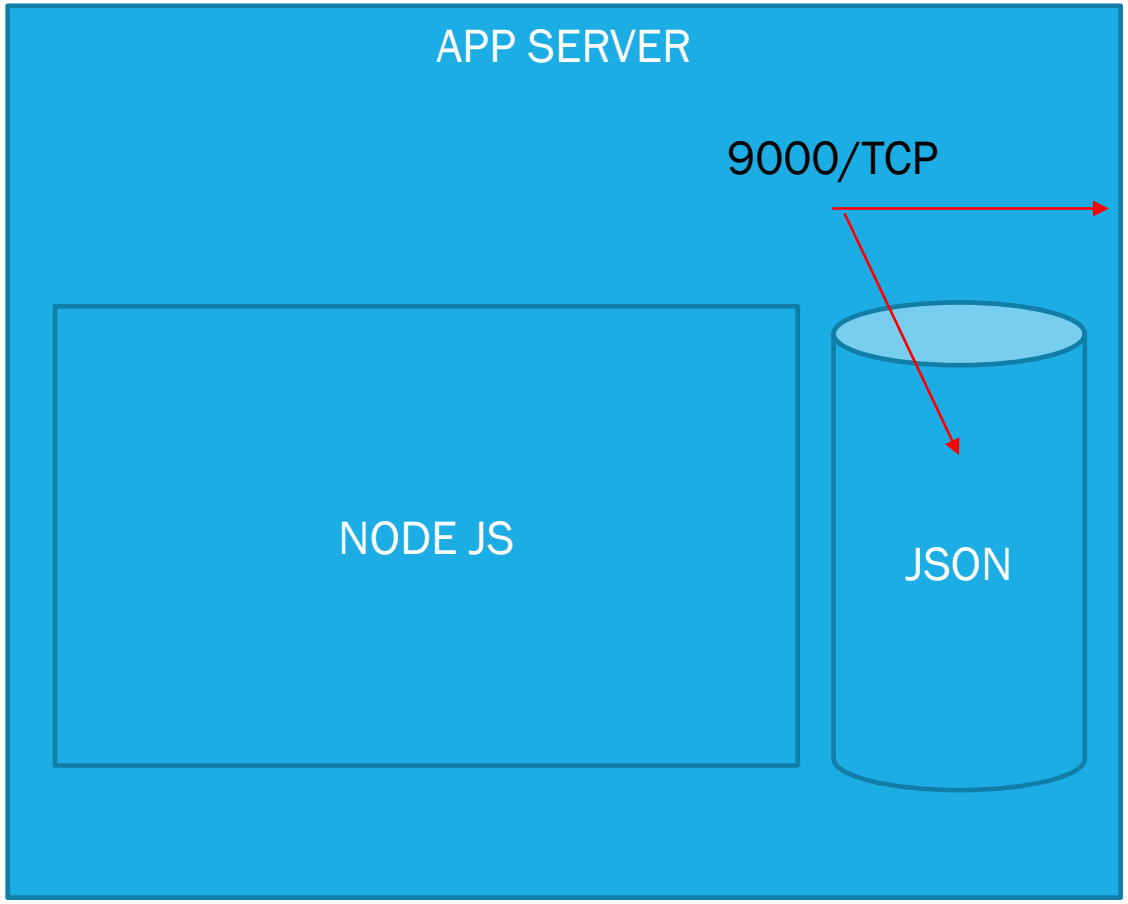
DOCKER VS KUBERNETES

DOCKER	KUEBRNETES
Use case: 1.Portable Applications across cloud (Image Portability) 2. High performance for deployment (Light weight Containers) 3. Possibility of Vertical Scaling (Container as Deployment) 4. Abstraction of Infra from Applications	Cluster of Container runtimes 1. Scaling Horizontal/Vertical (Ha) 2. **Rollout (Ha)** 3. Customizable Framework (Any Container runtime) 4. Self Heal (Recovery)/Orchestrator as a service
Dynamic Resource management	Scale /REPLICATE
Service Definition , Service Deployment as Containers	Scale Services , Secure Services , Rollout Services
Development (Dev, QA)	OPERATIONS

CONSIDERATIONS ON PORT FORWARDING…

STATIC PORT FORWARDING	DYNAMIC PORT FORWARDING
PORT FORWARDED SERVICE (DESITINATION PORT) _ CONSTANT FOR SERVICE (CONTAINER)	Dockerd will assign free at start/stop of container
Devops – Manual Maintenance → Conflict of Duplicate Ports should be managed by manually	dockerd
Recommended tobe in Forwarded between 1 – 30000	Greater than 30000 (32767 >)
→ -p Hp:CP → -p 8000:80	**Service Name** /Cluster Support → -P

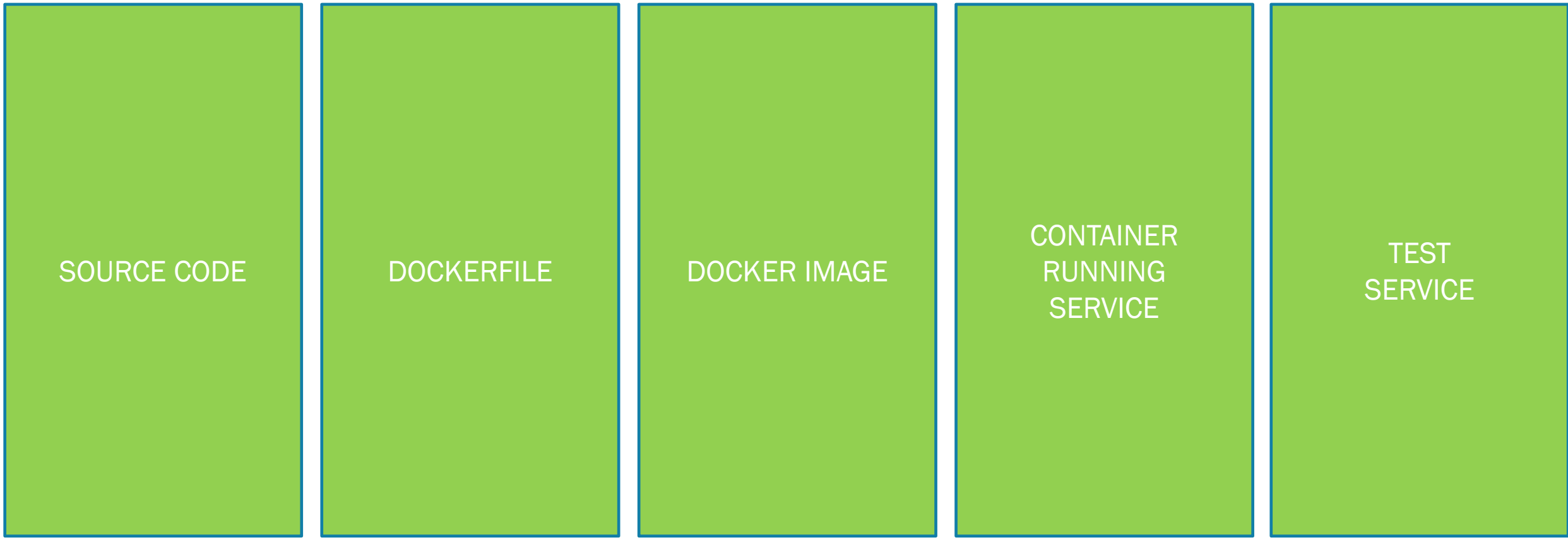
USE CASE - REST API –MICROSERVICES STRATEGY



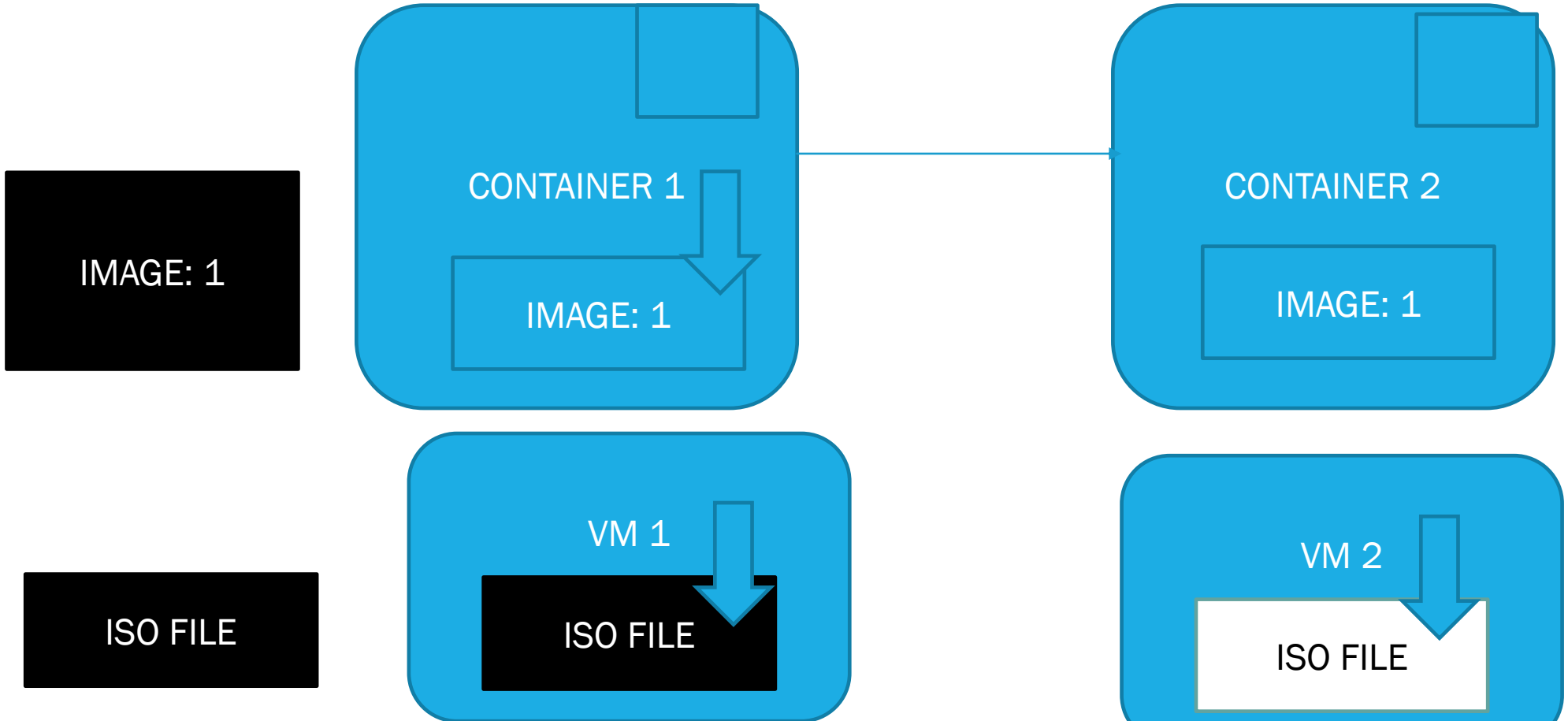
Node
Code.js
(Java Script Server Script)
Data.json
9000
==========
Body-parser
express

```
# REST API Implementation
from node
label maintainer dj@app-server.com
copy node.js /code-app/node.js
copy data.json /code-app/data.json
run npm install -y express body-parser
expose 9000
cmd node /code-app/node.js
```

IMPLEMENTATION – REST API



IMAGES AND CONTAINERS



SHARING IMAGES – PART 1

DATA CENTER	CLOUD
SERIALIZE Docker Image Object to File (TAR FILE) # docker save –o <object.tar> <docker_image>	# Oracle Object Store
Recover Images from TAR # docker load < <tarfile>	