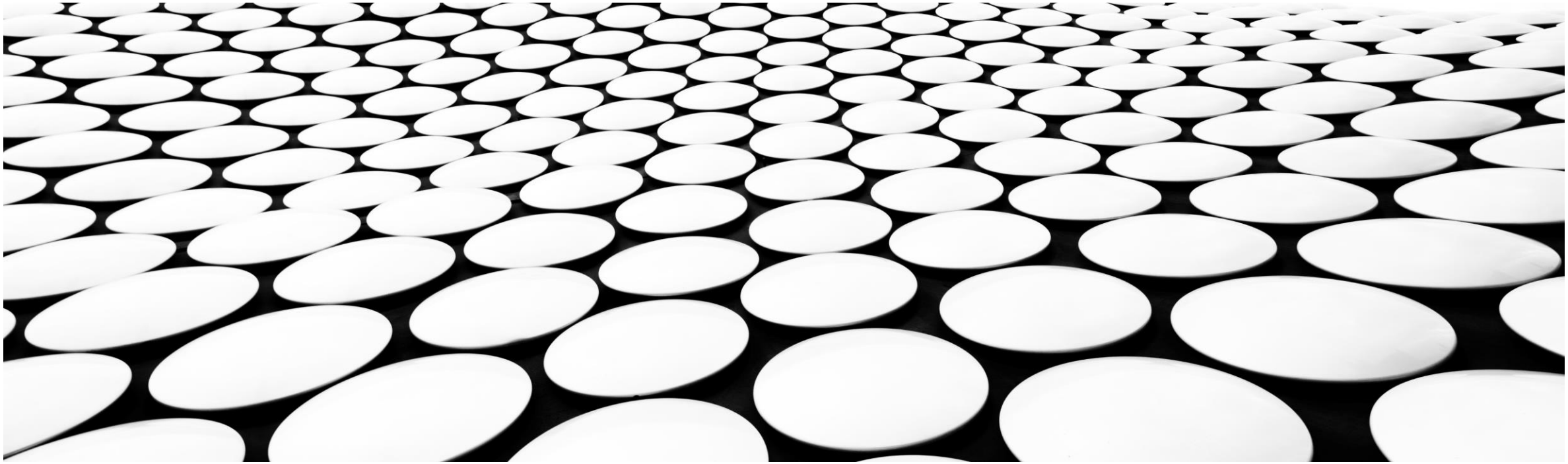
CONTAINER PIPELINES WITH ORACLE CLOUD

DJ (DHANANJAYAN)

DAY 2 – 30TH JUNE 2020



DAY 2

- Manage Containers
- Check Logs /Troubleshooting
- Dockerfile
- Expose Services
- Use Case – MSA Object
- Use Case – Database as Service
- Health of Service
- Port Forwarding Service
- Monitoring Services ?
- Share images to HUB, OCIR and Data Center

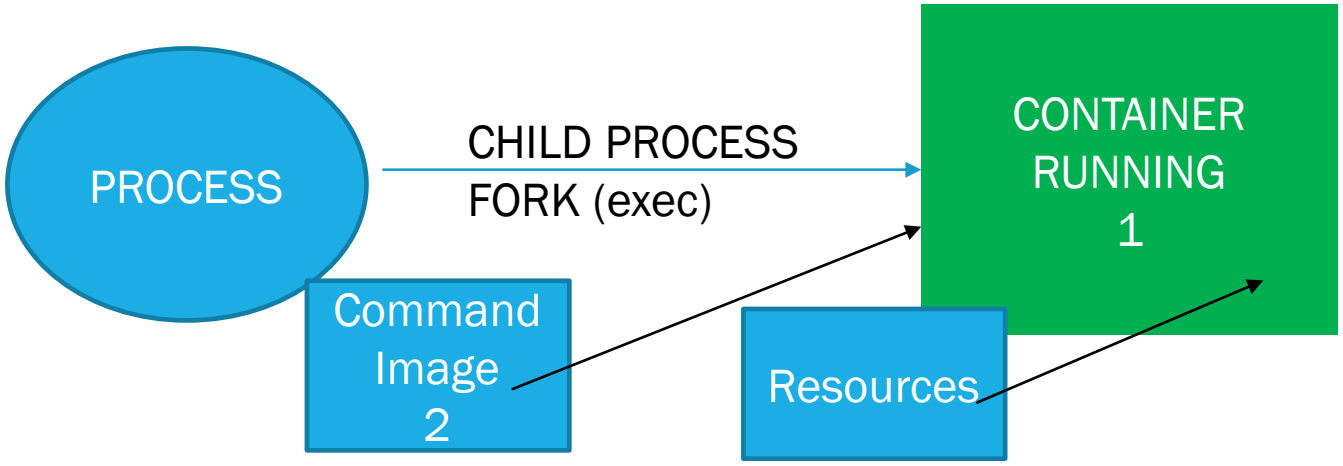
MANAGE CONTAINERS

- # docker start <container ID > or <container name>
  - `expression (back quote operator)` → UNIX
  - $ (expression)
- # -q (ids of the command)
- Extract from JSON
- → {{ Interpoliation}}
- Case Sensitive , → “.” Attribute Operator
- {{.State.Status}}
- #docker inspect –f “{{ }}”

LINUX CONTAINERS

- File for Image (Dockerfile)– Base Image
- /etc/resolv.conf → Nameserver file.
- /etc/hosts → information about host

Read anything from Container
Read a Service/File
Monitor in Container
Explore Process in container.



RUNNING CONTAINER

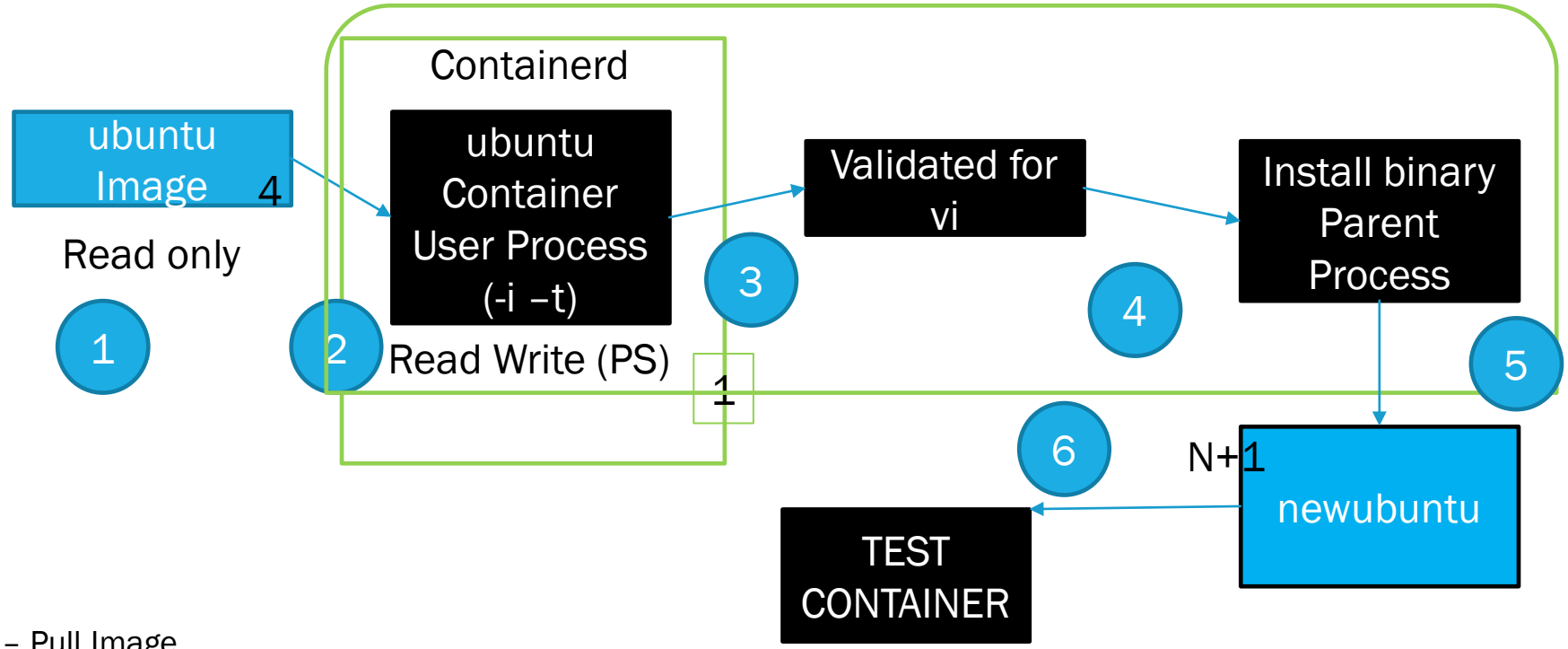
PARENT PROCESS	FORKED PROCESS
Administer resources for Container Manage the Child Process of container Configure env.variables of container	Read a Log File Monitor a Process Report a Service File Systems.
Parent ps → Server Process → Listener to connect to Parent ps → user process → (bash)	#exec
# attach	Child process .. Exit – Context lies only to child process
# --detach-keys, impact the process within container	Parent process still continues to run

CUSTOM IMAGES

Container → Image	Dockerfile	VM → DIS
# docker commit	# docker commit as below SOP (detailed document..step..)	# rancher vm
Limited Layers, No API Support Synchronous, Manual Deletion Not supported by Automation	Choice of Layers (publisher) API Support → Timing Asynchronous , background Automation Tools, CNCF, Automatically GC	# not approved by cncf # not automated # cost

CUSTOM IMAGE – STAGE 1

Entire CFS → Layer ?
Manually delete Containers
No API Support
Not supported by all cloud
Not supported by Automation



1 – Pull Image
2 – Create Container and start
3 – Exec into container and verify vi ? → Vi not found
4 – Parent Process – attach by detach keys → Install vi – apt-get install vim
5 – Container to Docker Image (ubuntu + vi) → docker commit container image-name
6 - pull, create cont, started, attach … commands.. ..rm

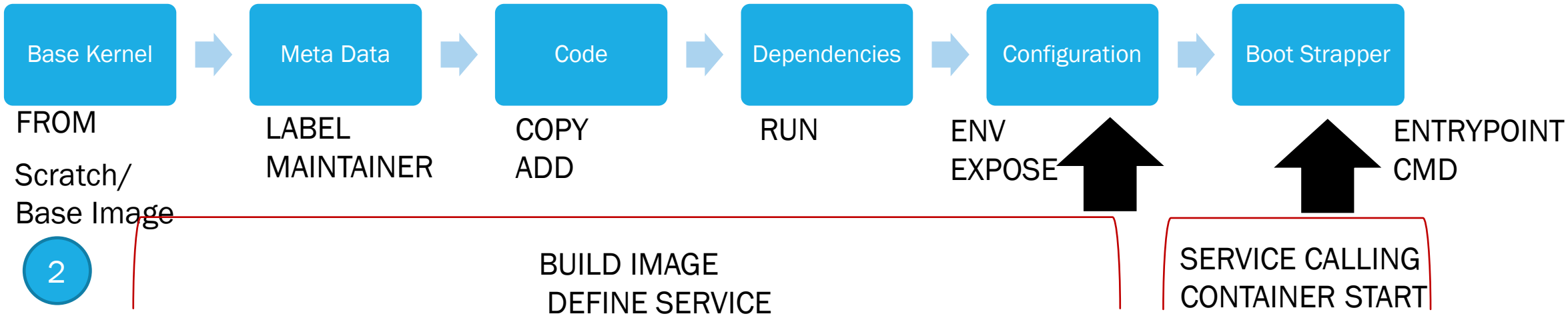
DOCKER IMAGE

- MONOLOTHIC
- SERVICE ORINETED
- MICRO SERVICE ARCHITECTURE

1

- PACKAGING YOUR APPLICATION
  - MAKEFILE
  - ARCHIVE (JAR/TAR) -- RUNTIME

2



Dockerfile

Automation File to Build Docker Images
Choice of Layers
Asynchronous , CNCF – CSP will abide by rule.

docker build –t <image> -f <Docker_file> <CONTEXT_ROOT>
WEB_CONTEXT (CONTEXT_ROOT)
Image name (-t) , Lowercase
Dockerfile – custom Dockerfile (-f)
Context ROOT →Directory path for dependencies for Dockerfile

USE CASE – SHELL APPLICATION

- SHELL APPLICATION → COMMAND LINE PARAMETER
- ARCHITECTURE – MONO
- PACKAGING is REQUIRED RUNTIME (BASH)

- DOCKERFILE
- BUILD DOCKERFILE (newubuntu:1)

- ENTRYPOINT [“sh”, “/code/Sample.sh”]
- CMD [“/etc/hosts”]

- INVALID FILENAME
- VALID FILENAME
- NO PARAMETER FILENAME

IMAGE → APPLICATION (SHELL)
Source Code – Path ?

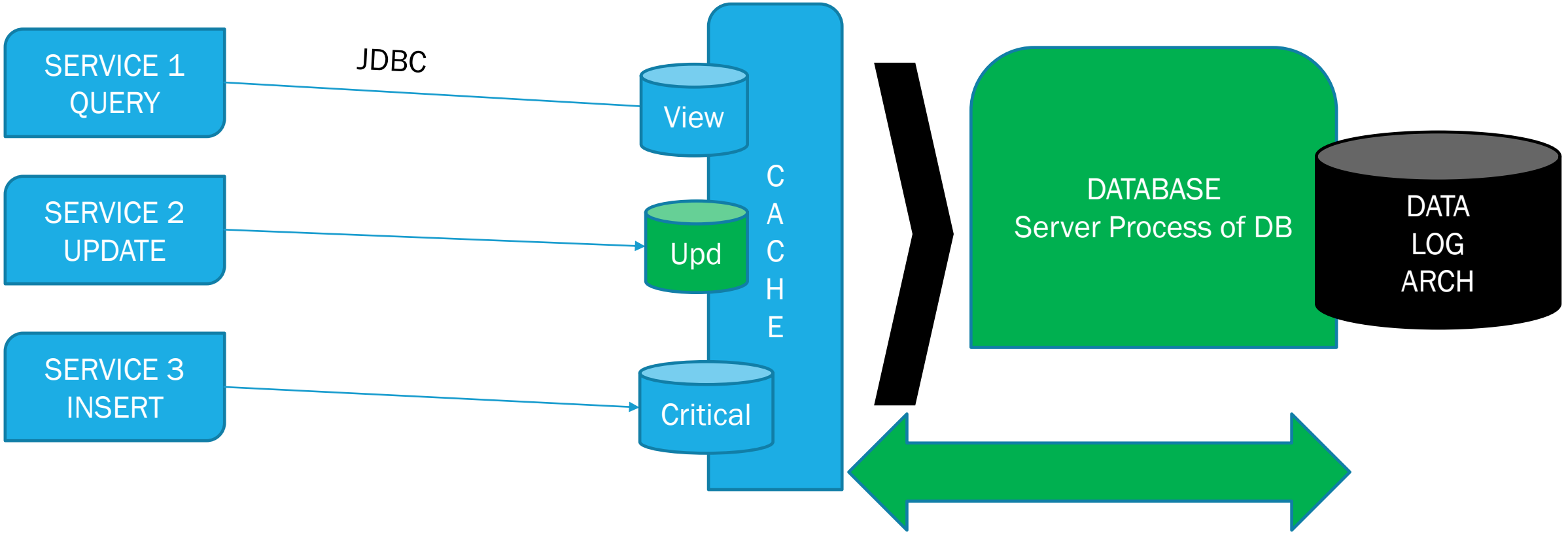
Flexibility:
$0 → FIXED
$1 → Paramter 1

Fixed Boot Strapper
Sh /code/Sample.sh
Variable Parameter
/etc/hosts

COMMAND LINE PARAMTER
[LIST OF ARGS]

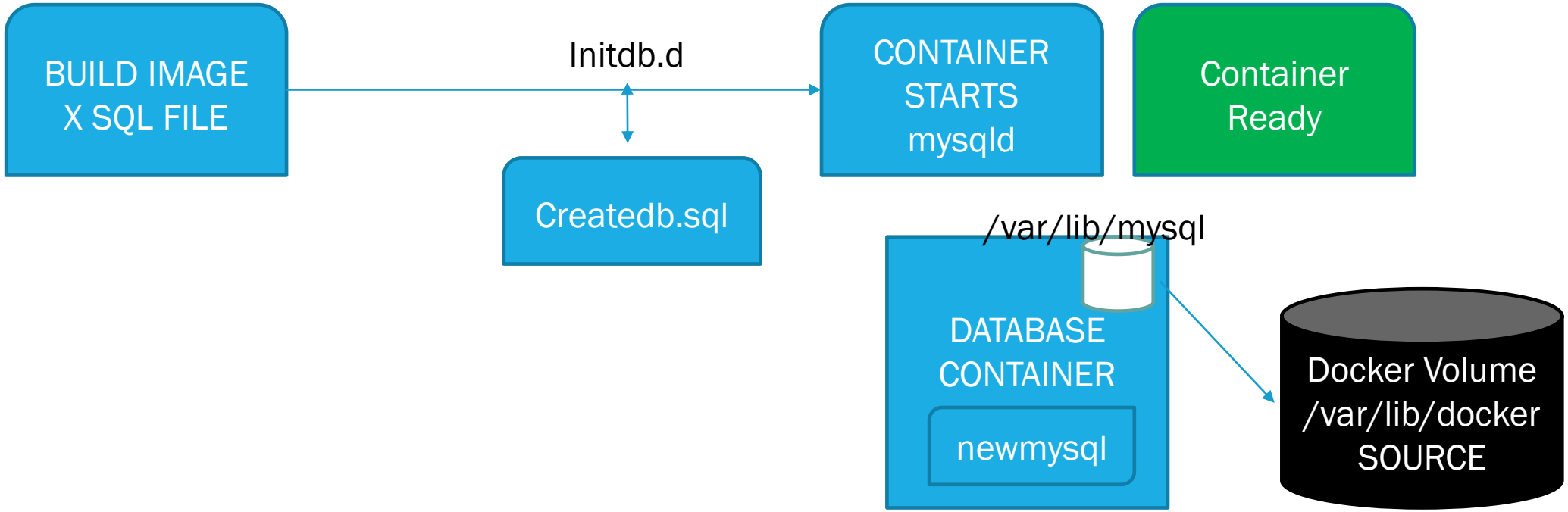
USE CASE (2) : DATABASE

- MICROSERVICES AND DATABASES

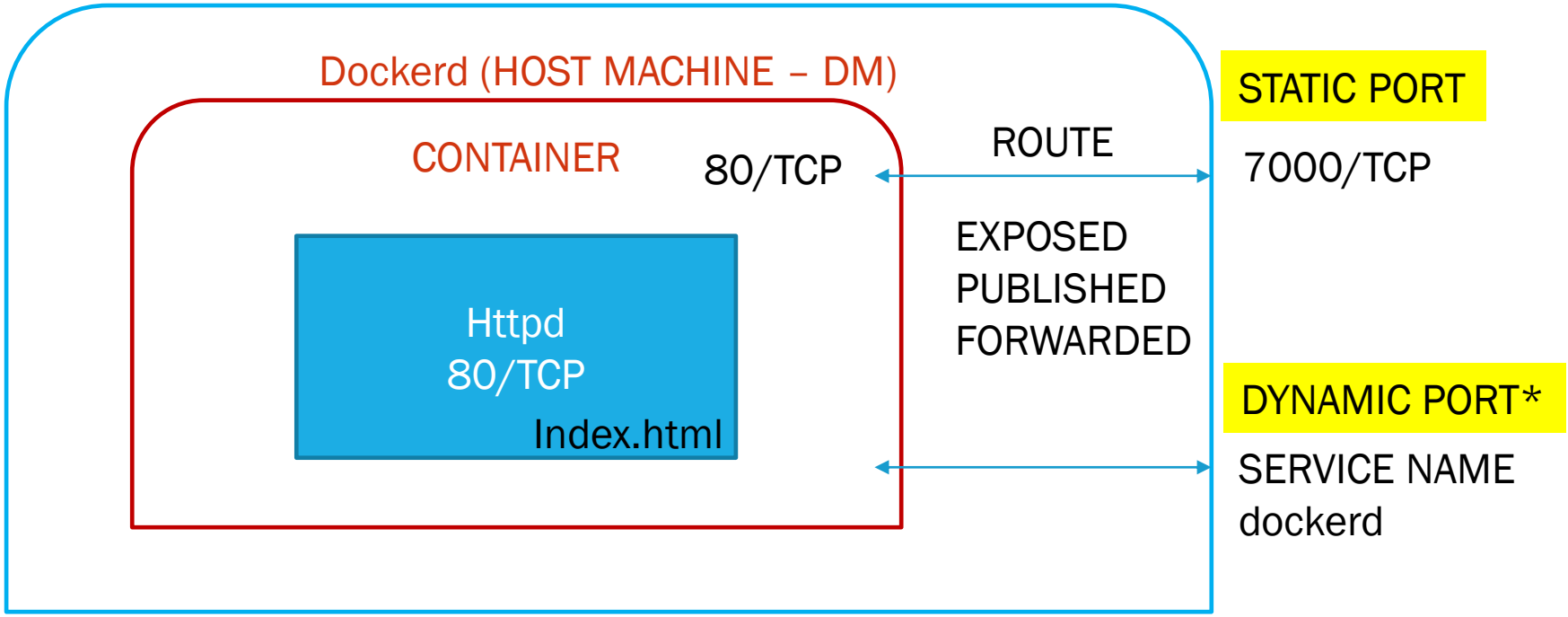


USE CASE : DATABASE CONTAINER

- MYSQL_ROOT_PASSWORD → TO START A DATABASE
- PATH DIRECTORY OF INITDB.D → /docker-entrypoint-initdb.d



SERVER PROCESS -- INTERACTION



Synchronous Call
REST API Call
1-1 Communication
Request – Response
Format – Developer
Indefinitely Block
Timeout=Output
100 – Information
200-300 – OK
300 – Circuit
400 – End point
500 - Server

ASynchronous Call
MQ Calls
0 - # Communication
Request – Response
Format – Developer
Timeout=Output
Pub - Subscriber

FORWARDS SERVICES THROUGH RPC

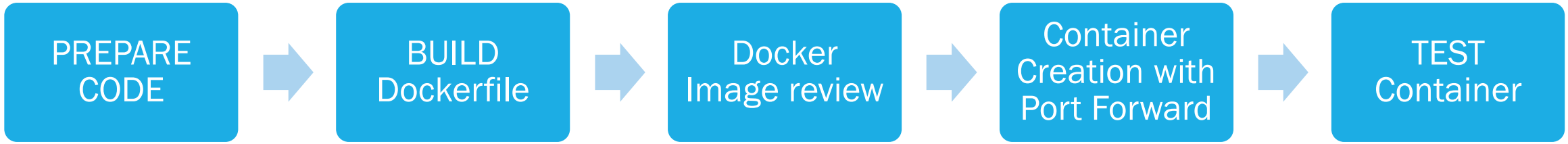
STATIC PORT FWD	DYNAMIC PORT FWD
Forwarded Port to Host – Decided by Ops (Devops)	Forwarded port to Host – Decided by dockerd
LESS THAN 30,000	32768 – 35999
ACCESS by port number	Access by Service Name
Knowledge Management – Allocate Free port Iptables –t nat –L*	Automatic Sequence.
Start/Stop Container – Restart – Forwarded port Fixed	Forwarded port Fluctuate /Service Name
-p HP : CP -p 7000:80 –p 7001:100	-P

USE CASE : REST API (MICROSERVICES)

- Independent Service
  - Definition (Source Code)
  - Deployment (Container or VM or Host)
  - By Replication (SCALE)
  - Private Data Store
- PORT – 1 Primary (1 – Backup)
  - MULTIPLE END POINT
    - http://domain.abc.oracle.com:8000/abc --> End point → Redirect to specific module
- LISTENING PORT (Primary + Backup Port number)

NODE JS – Server Side Scripting
DATA Store (JSON)
9000/TCP
ListUsers → All users in JSON
Response → JSON

USE CASE : APP SERVER (REST API)



--node.js
--data.json

```
FROM node
LABEL MAINTAINER dj@appserver.com
COPY node.js /code/node.js
COPY data.json /code/data.json
RUN npm install -y express body-parser
#metadata for Docker image. Does not validate.
EXPOSE 9000
CMD node /code/node.js
```

```
# docker run...
```

USE CASE: HEALTH...

Container Running ?
Response for EndPoint ?
Format of Response

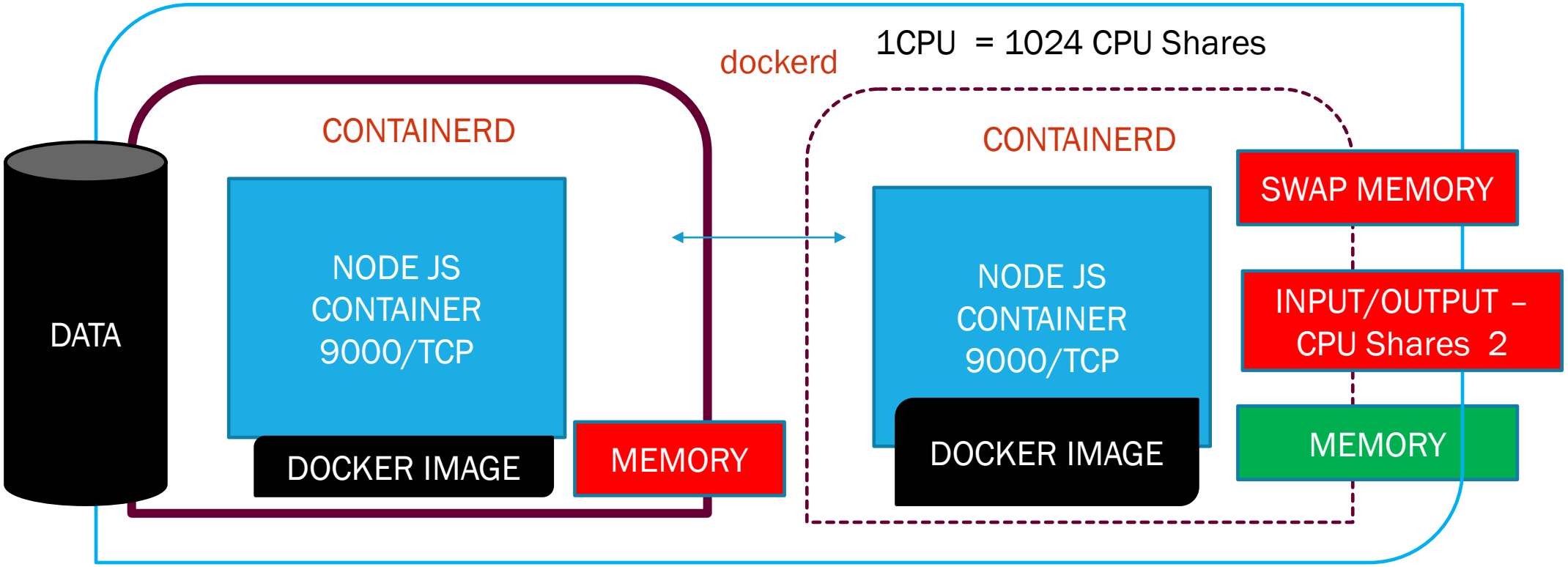
#curl for service
docker ps –a

Response time within Threshold
Proactively being tested
output of threshold...
Services is Healthy

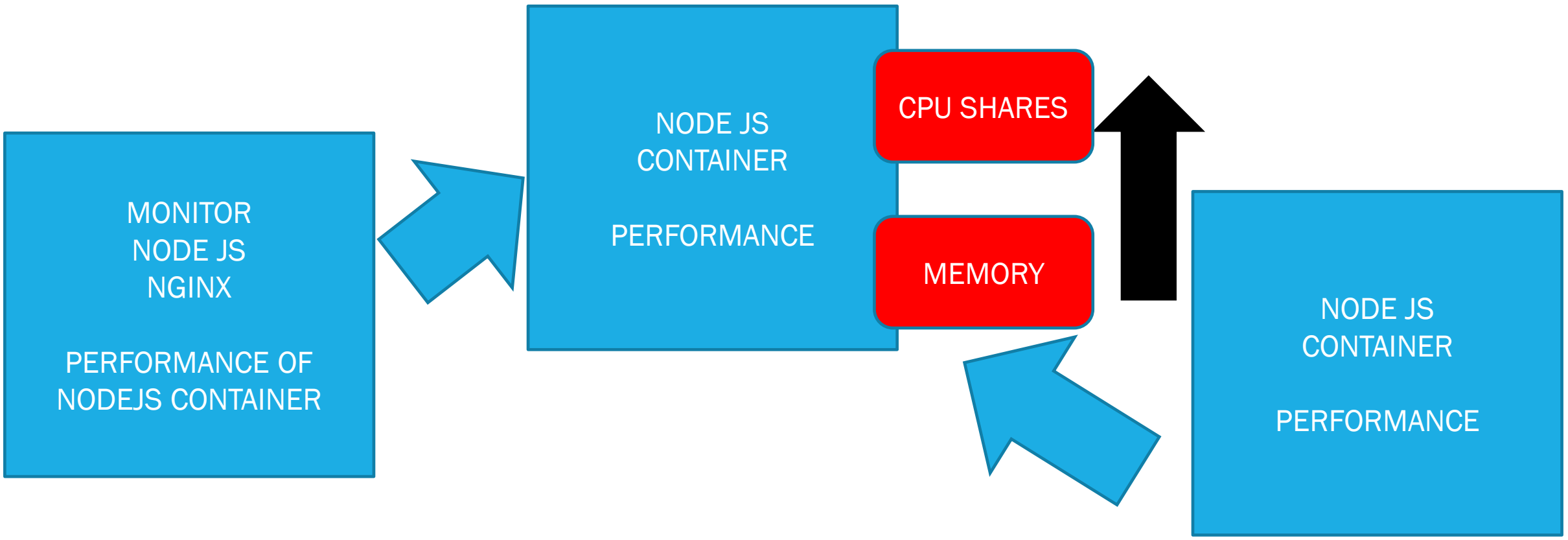
Infrastructure are resource within acceptance of subscriptions
Resources are Healthy

#docker update..

USE CASE:...MSA FOR INFRA (RESOURCES) HEALTH ?



USE CASE : DEPLOYMENT HEALTH (SIDE CAR DESIGN PATTERN)



SHARE IMAGE

SAVE it in OCI	Save it in HUB (Docker HUB)	Data Canter (Backup)
Tenancy – Share resources Tenancyname - ocuocictrng22 1. MFA Auth Token (Oauth Token) 2. SSO Repository: iad.ocir.io *use auth token as password * tenancy/username 3. Prepare the Image repository/tenancy/image:tag iad.ocir.io/ocuocictrng22/newubuntu01 4. # docker push the tagged image	Username /password – HUB 1. SSO #docker login password as hub-password (account created for day 1) 2 Prepare the image hub_username/image:tag 3 # docker push <preparedimage>	Image Object→ FILE (Serialization) 1. # docker save –o <...tar> <docker image>