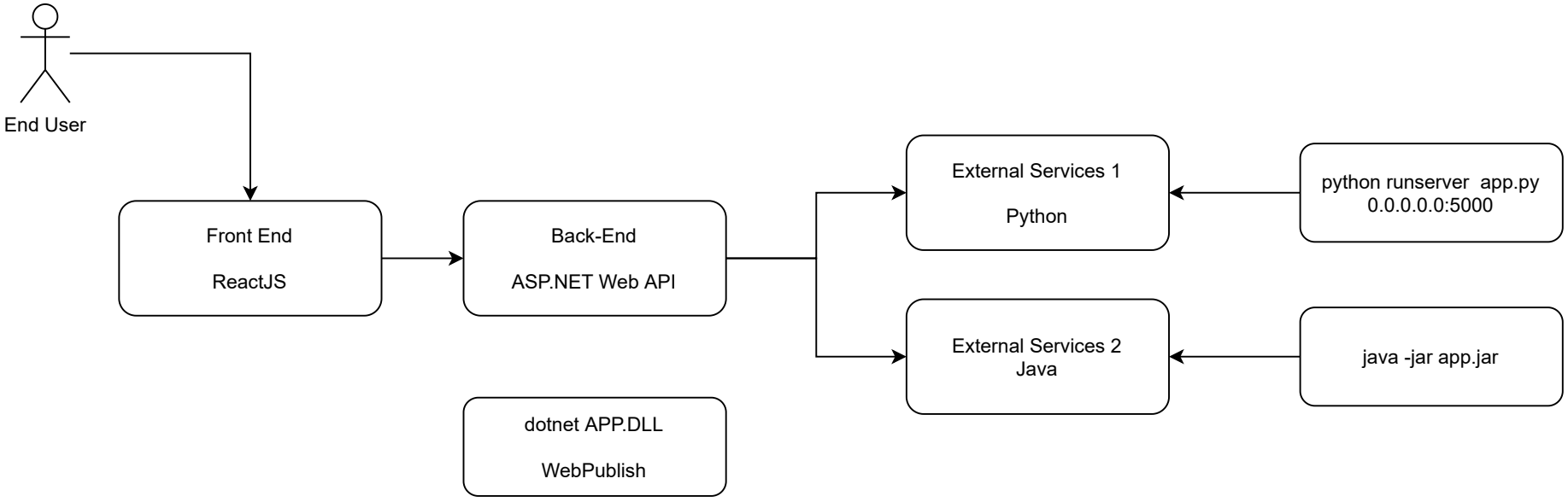
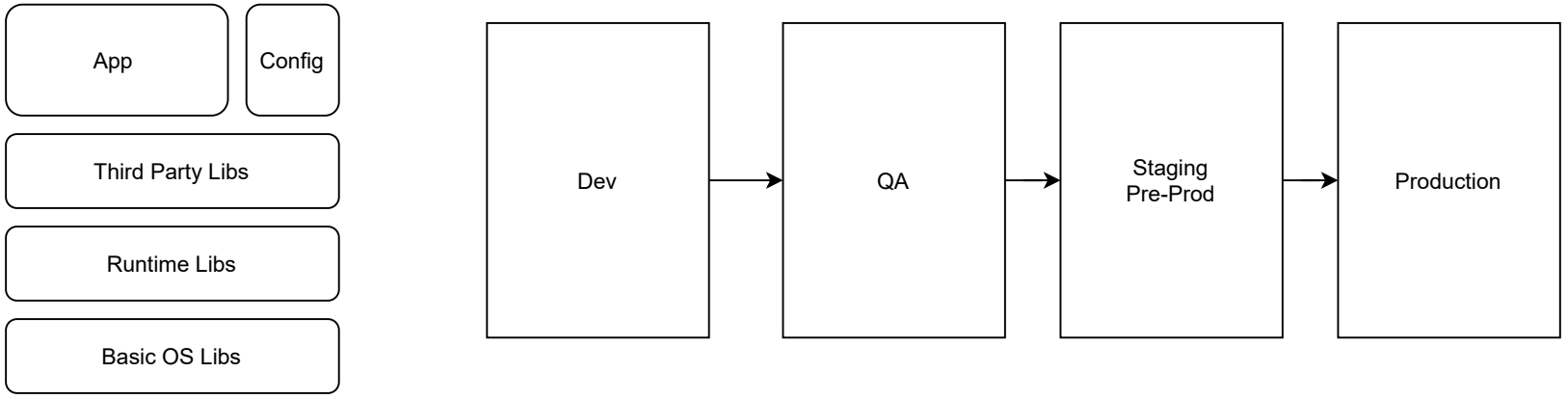
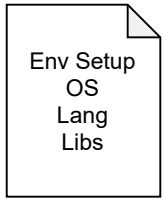
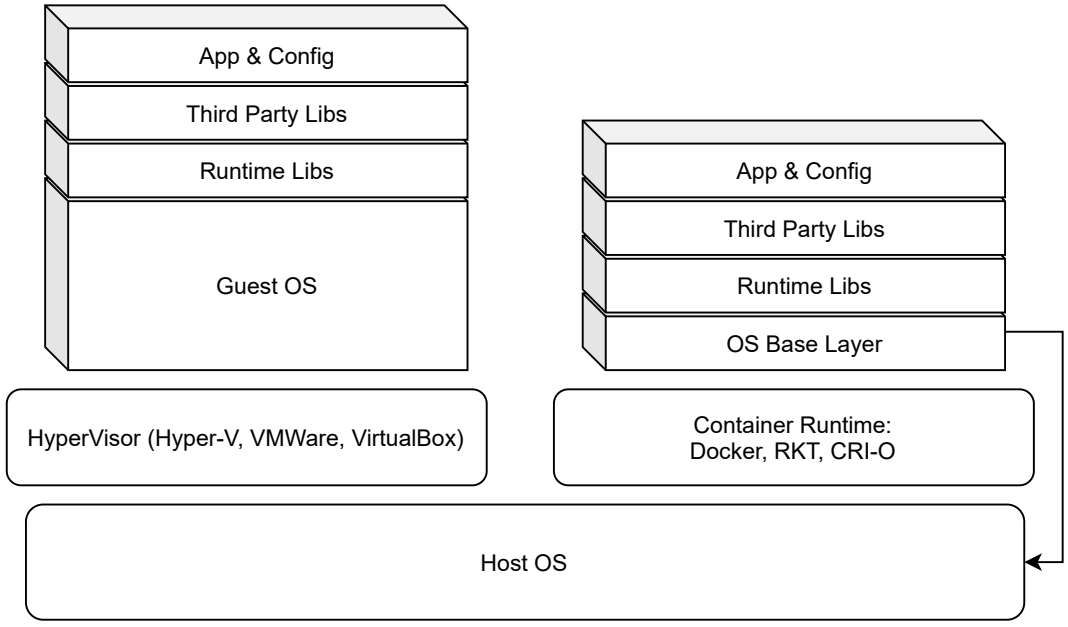
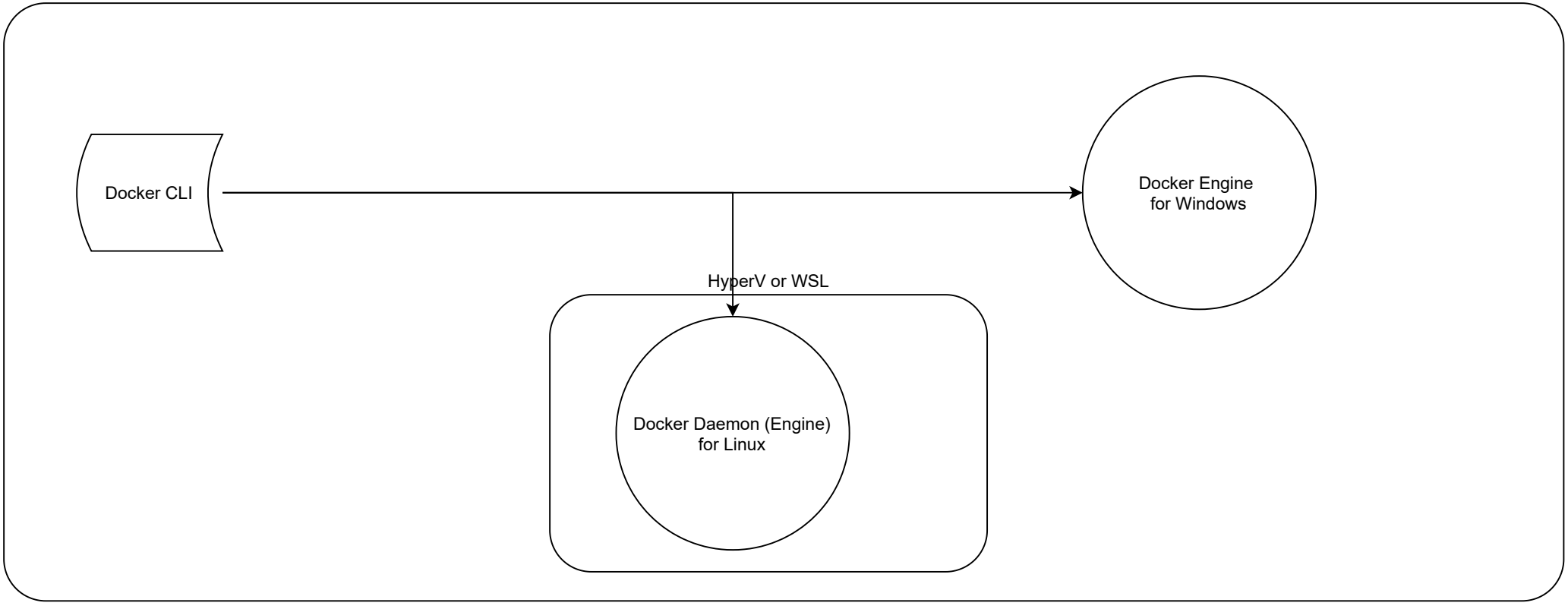
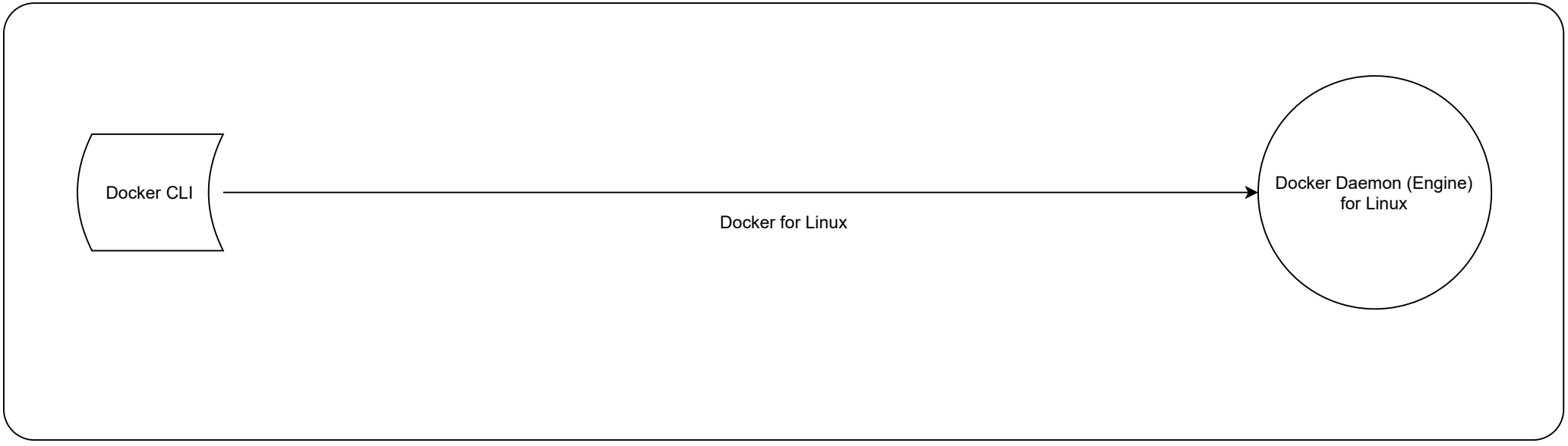

End User
Front End
ReactJS
Back-End
ASP.NET Web API
External Services 1
Python
python runserver app.py
0.0.0.0:5000
External Services 2
Java
java -jar app.jar
dotnet APP.DLL
WebPublish


App
Config
Third Party Libs
Runtime Libs
Basic OS Libs
Dev
QA
Staging
Pre-Prod
Production


Env Setup
OS
Lang
Libs


App & Config
Third Party Libs
Runtime Libs
Guest OS
App & Config
Third Party Libs
Runtime Libs
OS Base Layer
HyperVisor (Hyper-V, VMWare, VirtualBox)
Container Runtime:
Docker, RKT, CRI-O
Host OS


Docker CLI
Docker Engine
for Windows
HyperV or WSL
Docker Daemon (Engine)
for Linux


Docker CLI
Docker for Linux
Docker Daemon (Engine)
for Linux

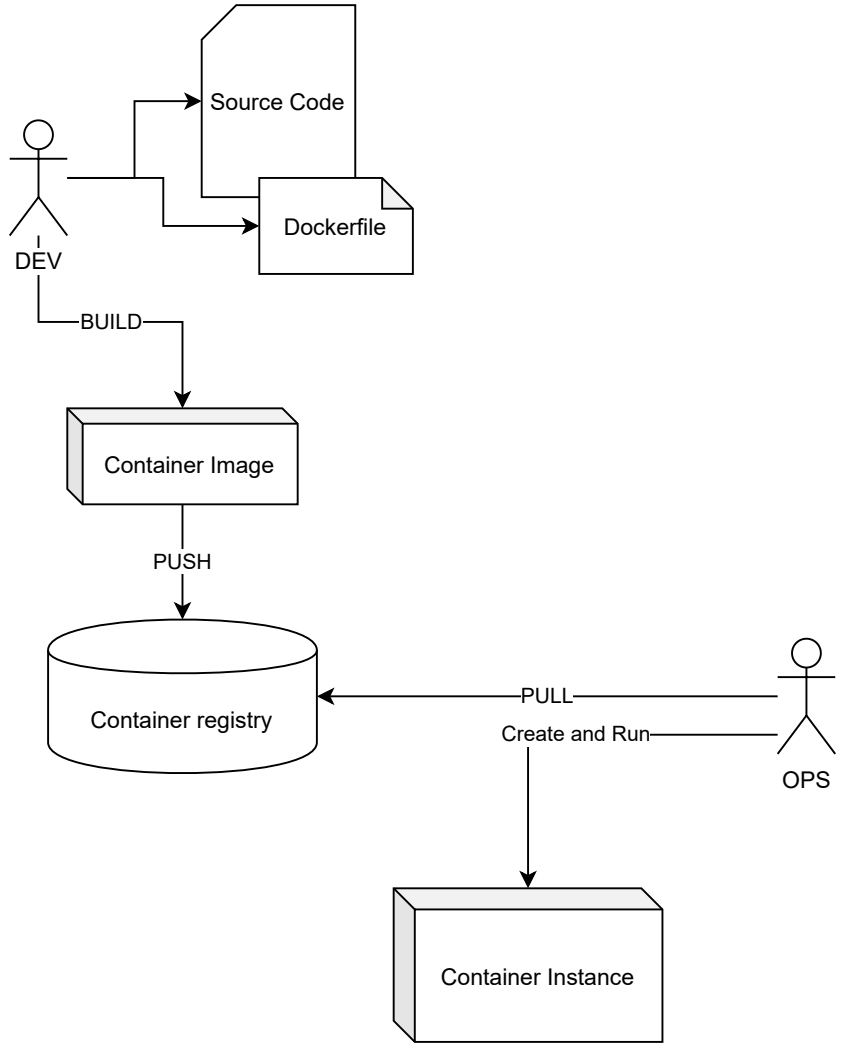
Container Instance
AKA
"Container"

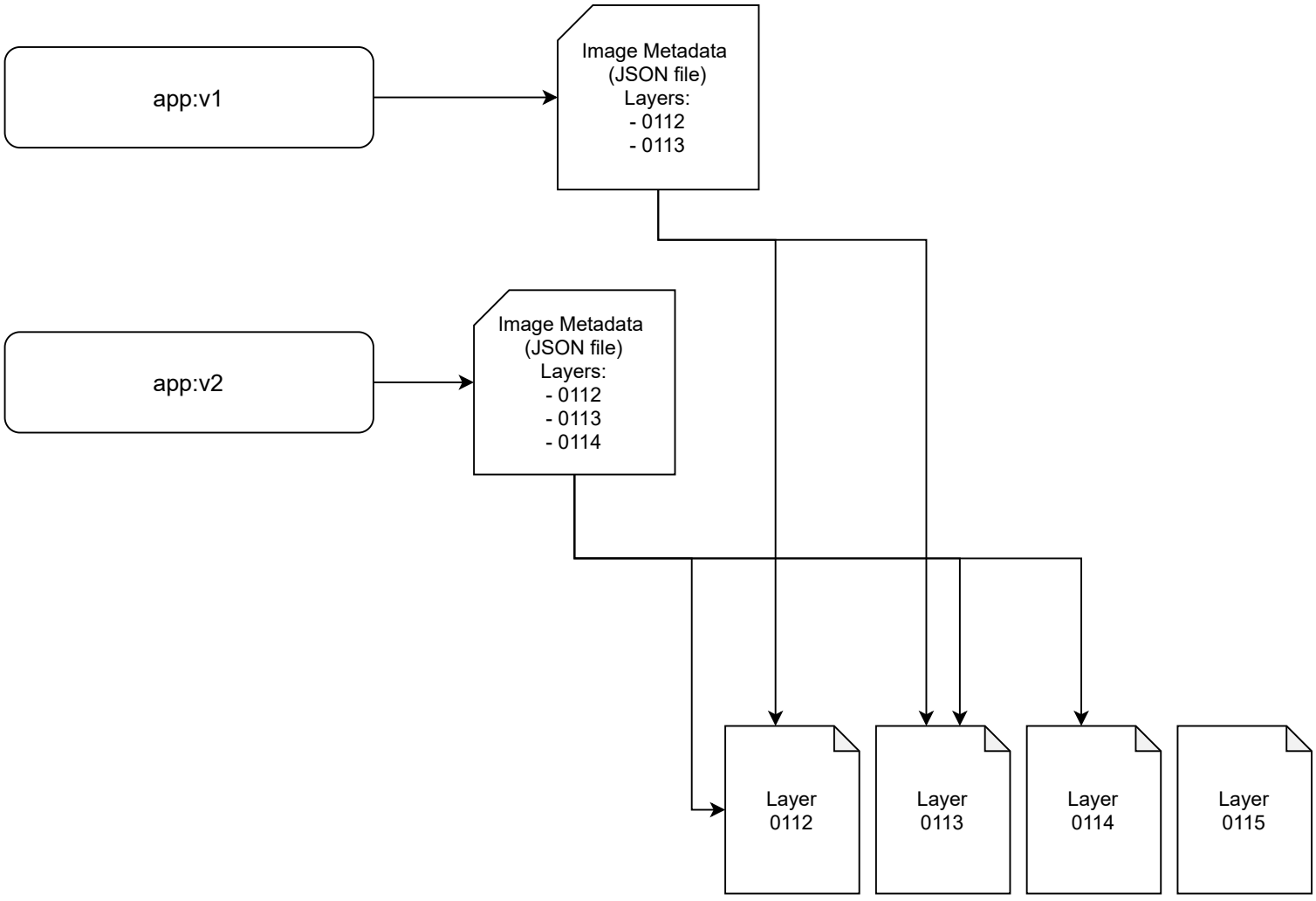
Container Image
AKA
"Image"

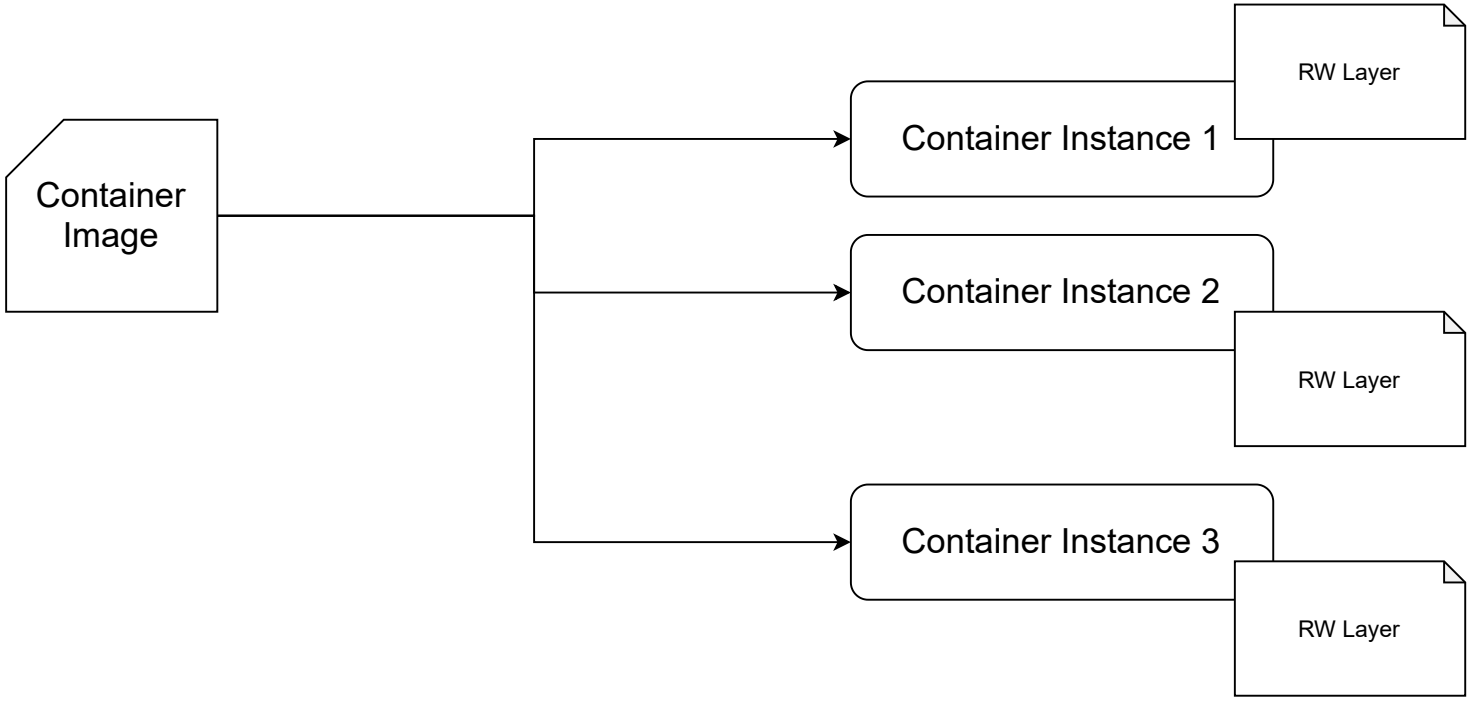
Container Volumes
AKA
"Volumes"

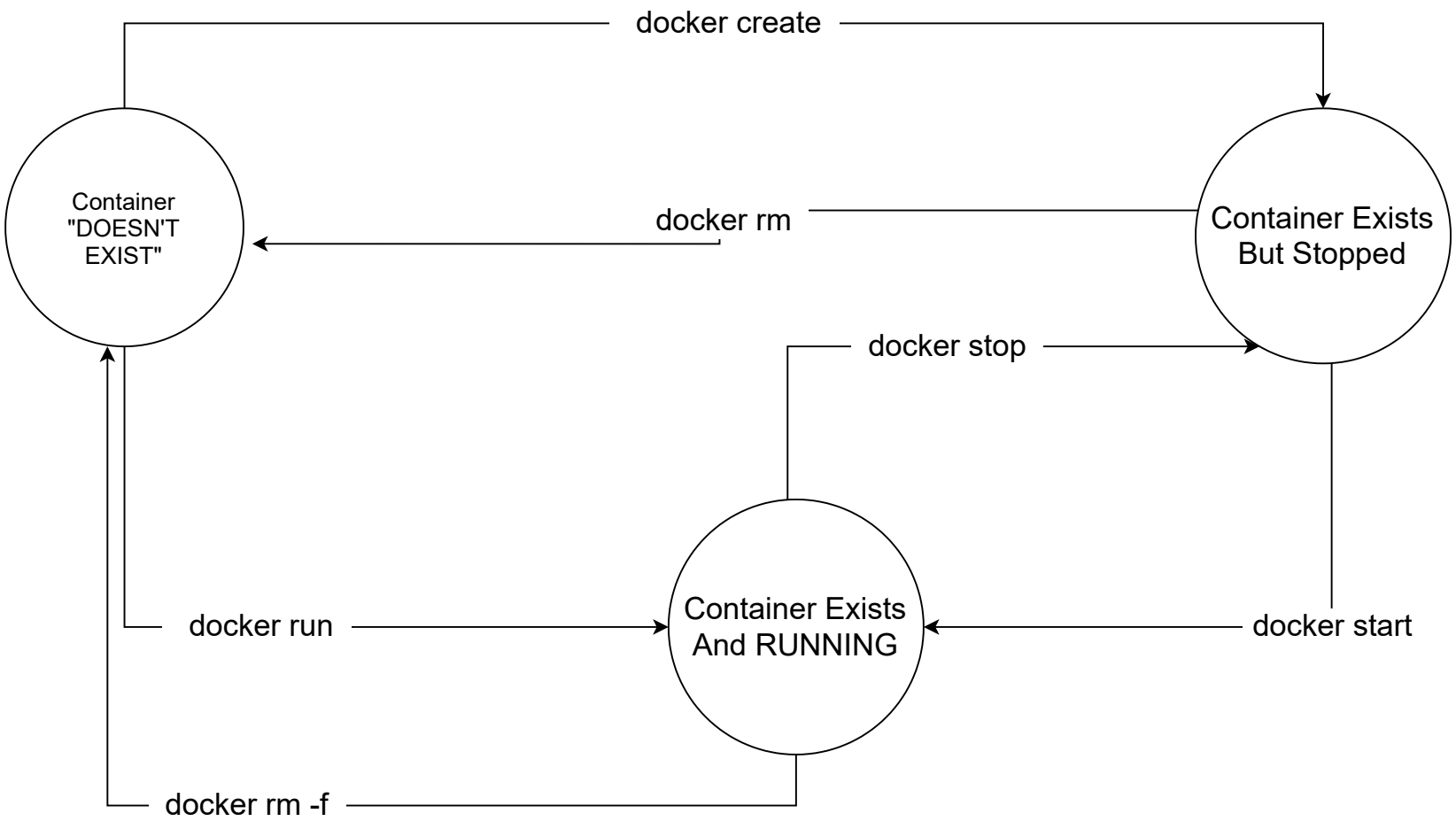
Container Networks

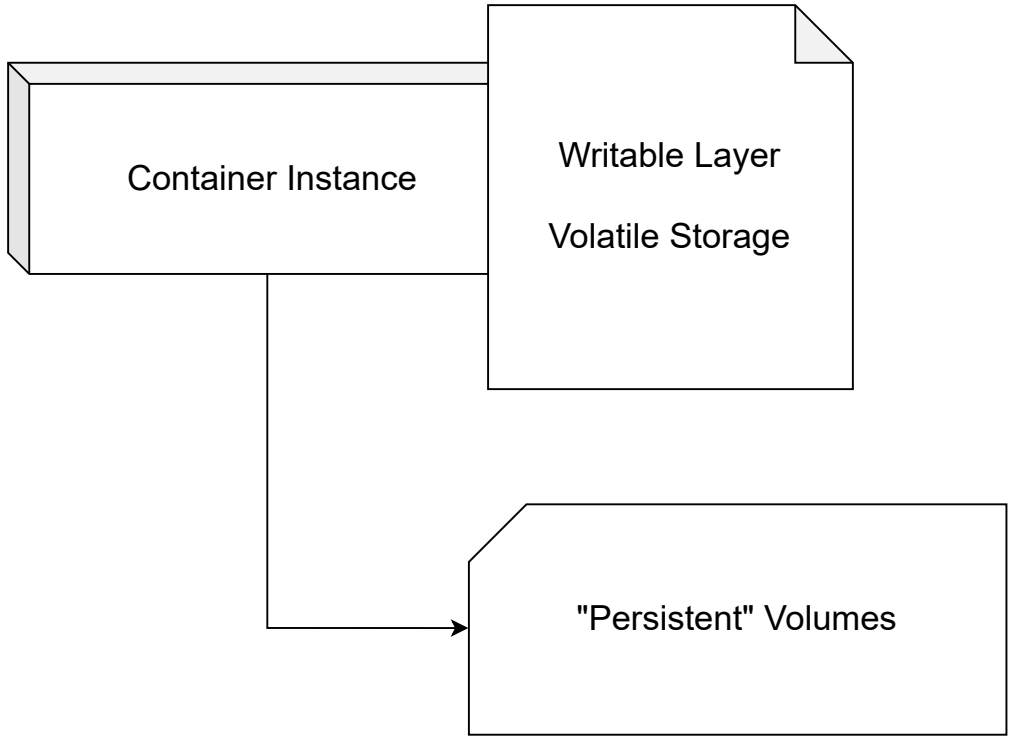
Container Registry

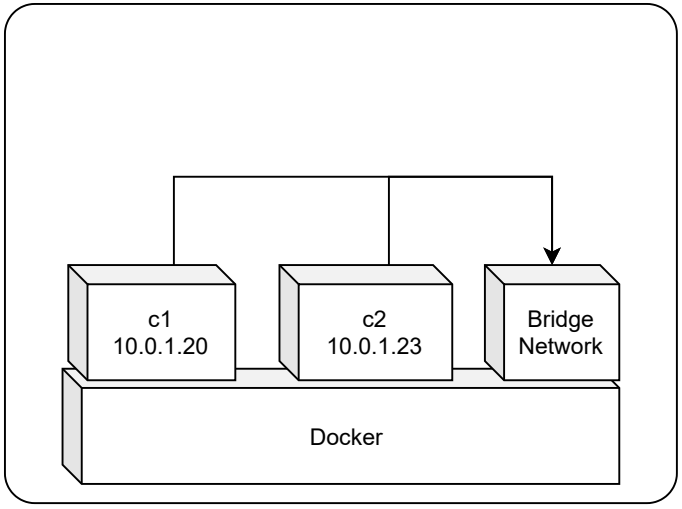


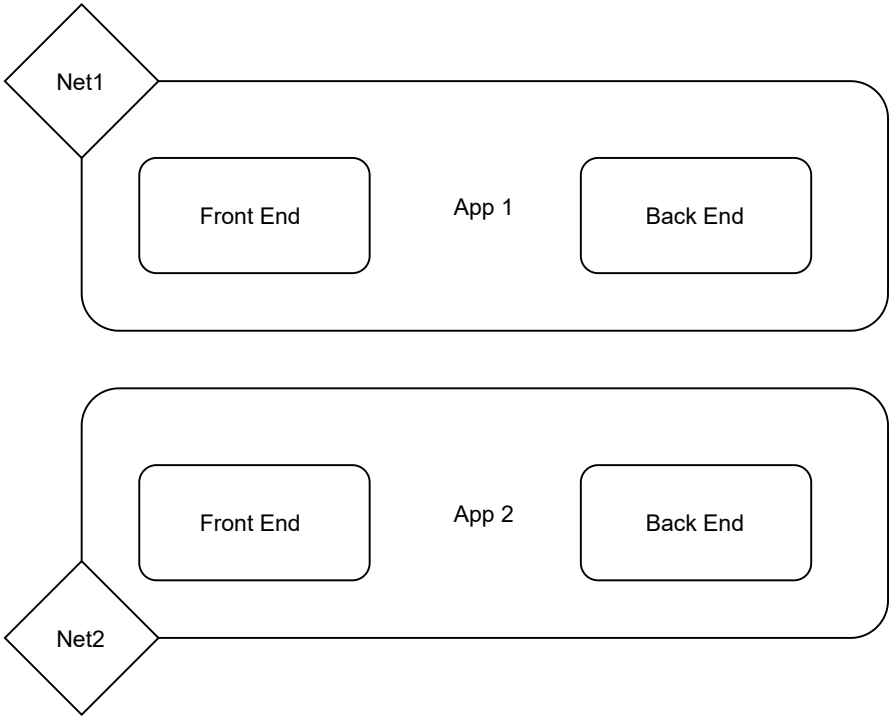

app:v1
Image Metadata
(JSON file)
Layers:
- 0112
- 0113
app:v2
Image Metadata
(JSON file)
Layers:
- 0112
- 0113
- 0114
Layer
0112
Layer
0113
Layer
0114
Layer
0115

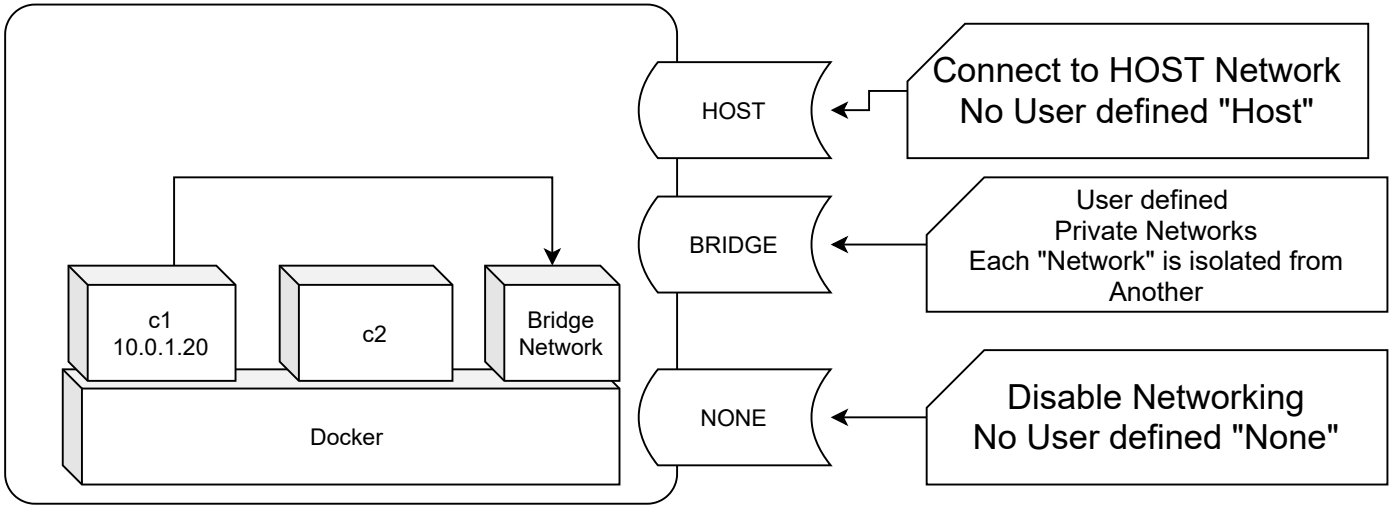

Container Image
Container Instance 1
Container Instance 2
Container Instance 3
RW Layer
RW Layer
RW Layer

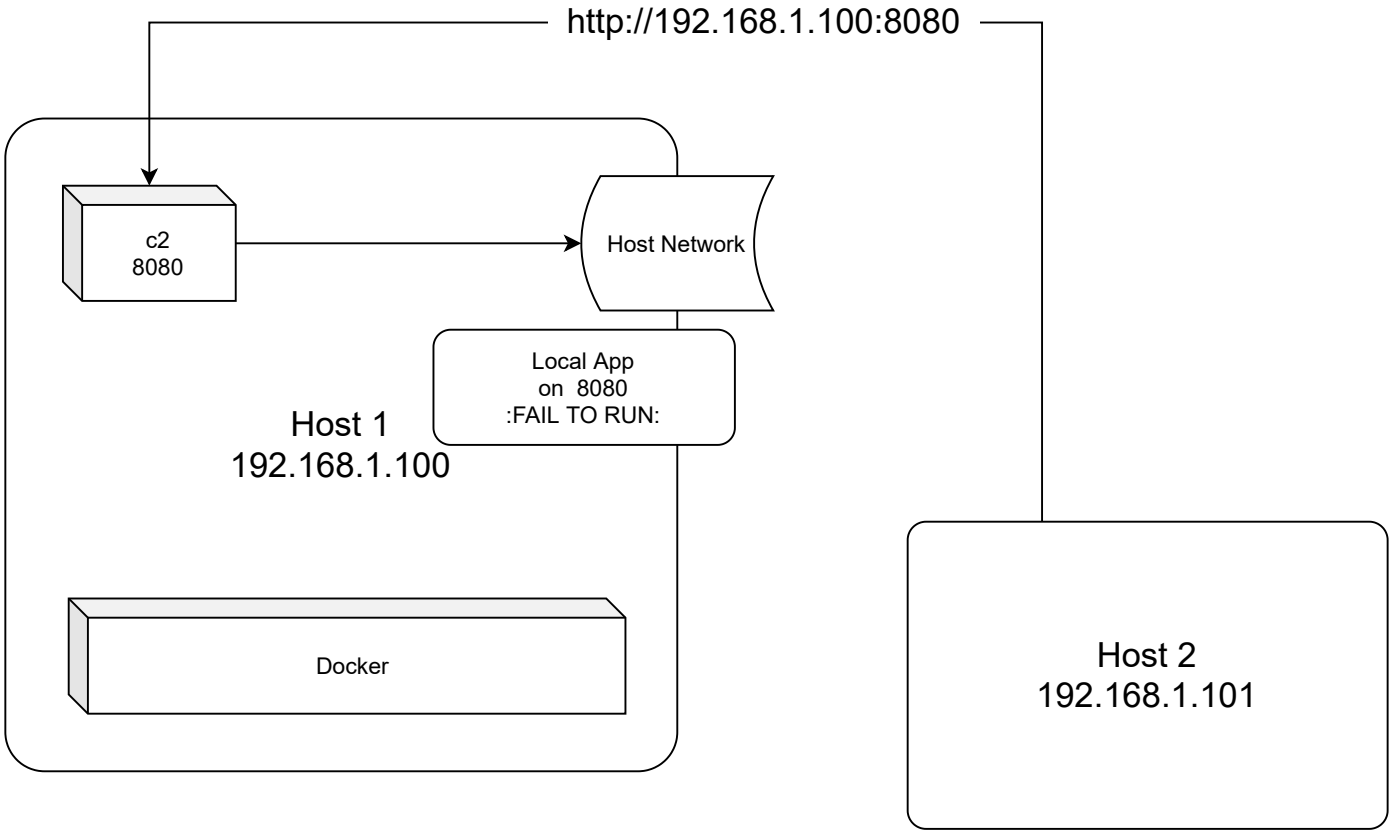

docker create
Container "DOESN'T EXIST"
docker rm
Container Exists But Stopped
docker stop
Container Exists And RUNNING
docker run
docker start
docker rm -f

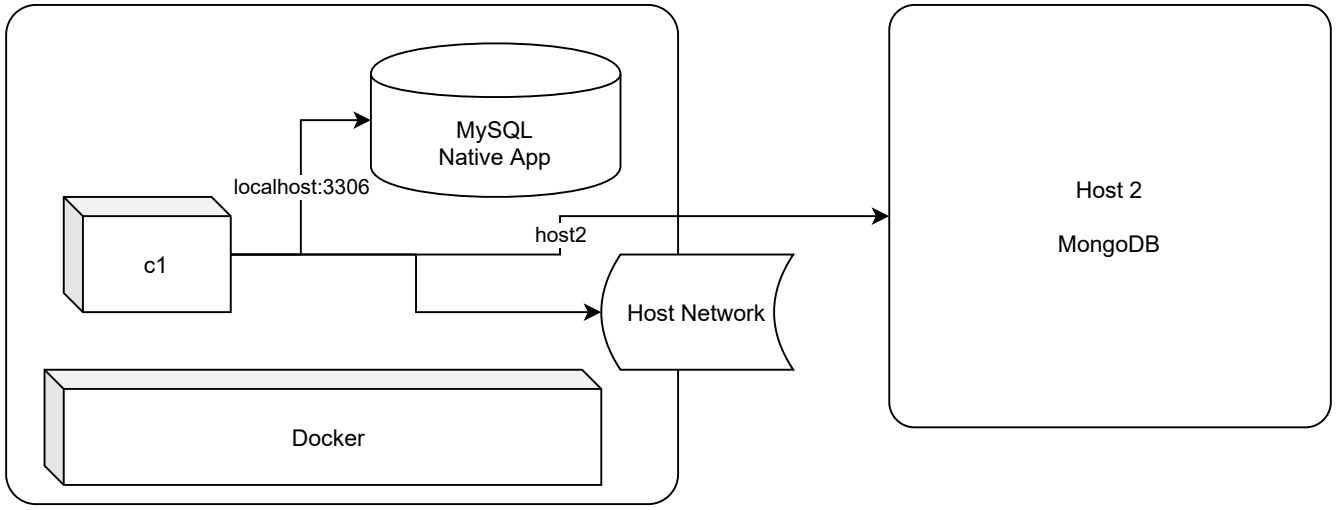

Container Instance
Writable Layer
Volatile Storage
"Persistent" Volumes

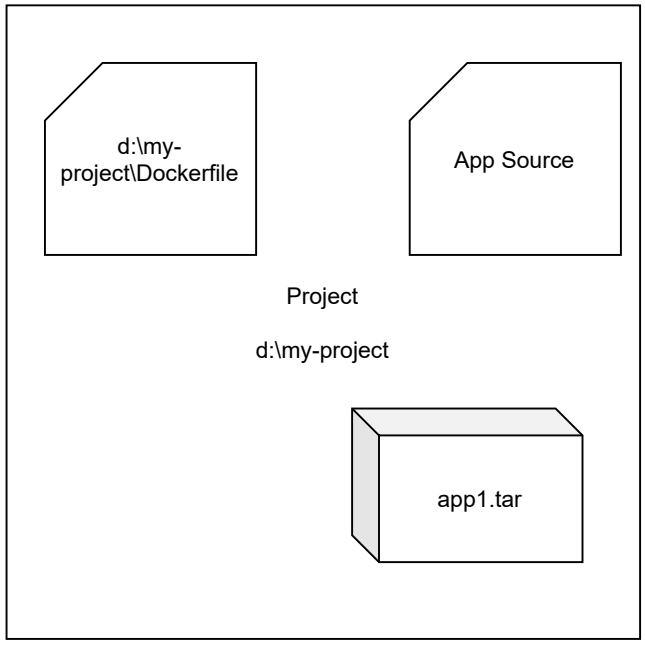

c1
10.0.1.20
c2
10.0.1.23
Bridge
Network
Docker


Net1
Front End
App 1
Back End
Front End
App 2
Back End
Net2


HOST
Connect to HOST Network
No User defined "Host"
BRIDGE
User defined
Private Networks
Each "Network" is isolated from
Another
c1
10.0.1.20
c2
Bridge
Network
Docker
NONE
Disable Networking
No User defined "None"


http://192.168.1.100:8080
c2
8080
Host Network
Local App
on 8080
:FAIL TO RUN:
Host 1
192.168.1.100
Docker
Host 2
192.168.1.101

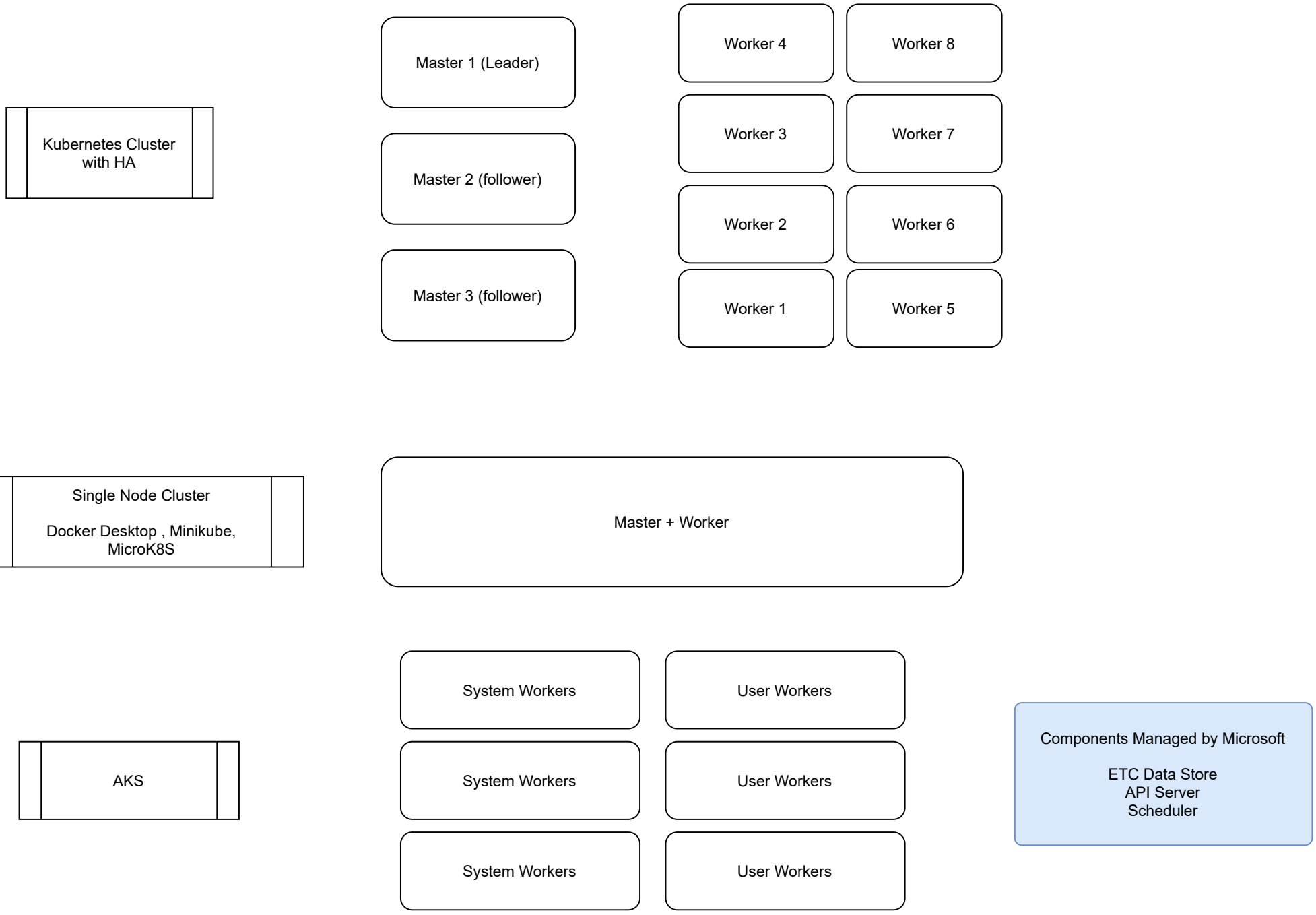

MySQL
Native App
localhost:3306
c1
host2
Host Network
Docker
Host 2
MongoDB

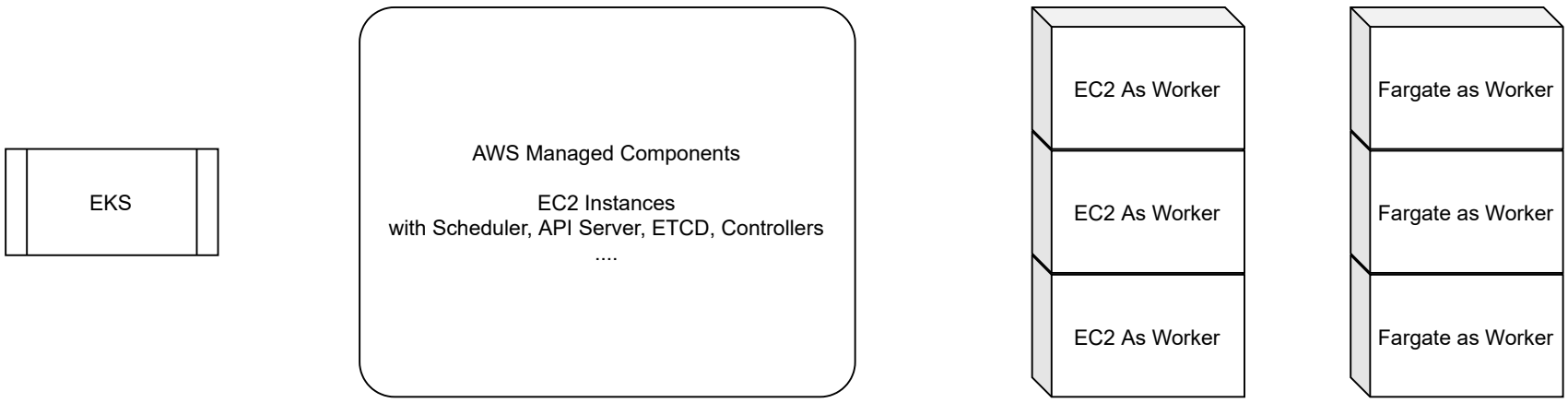


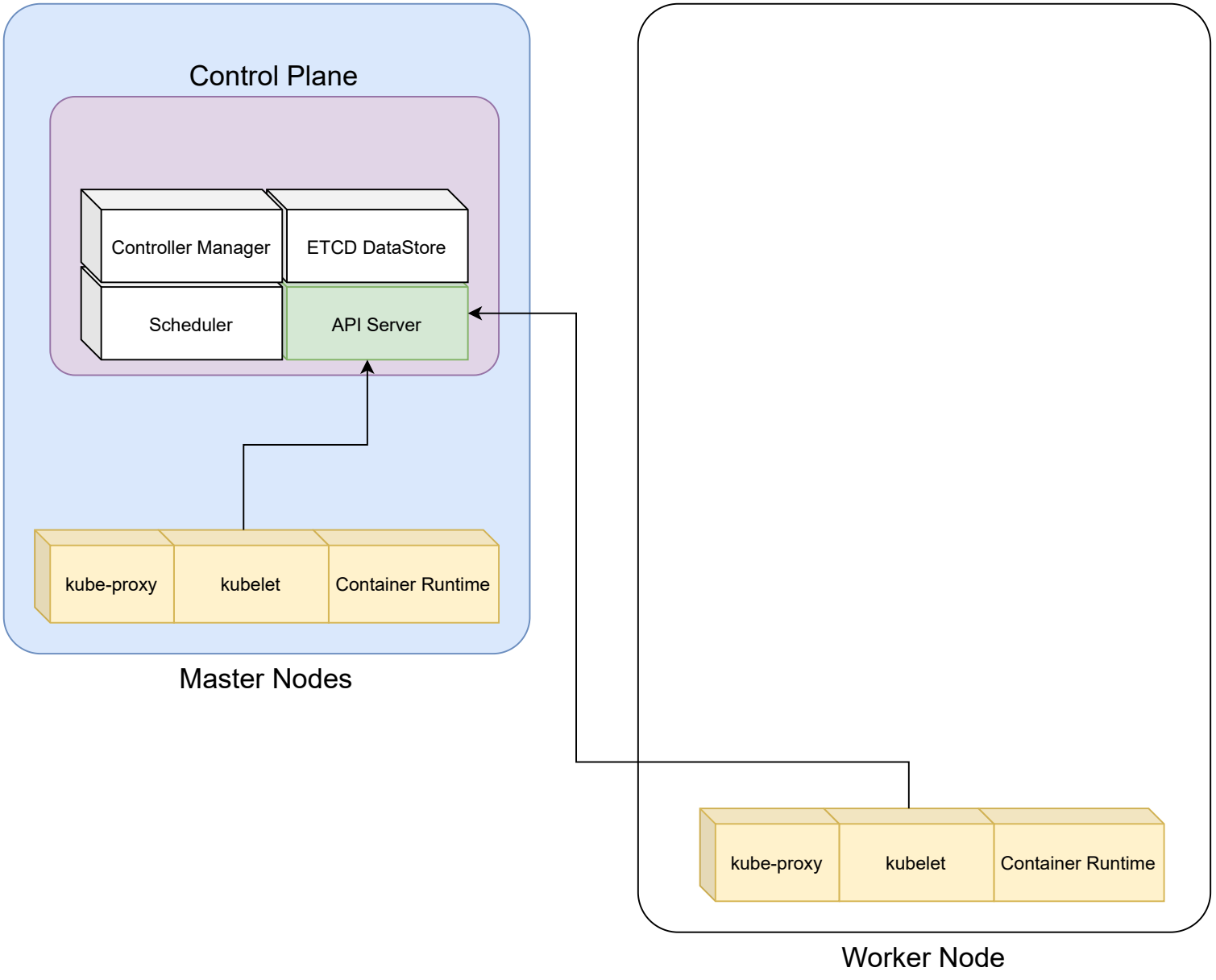

```
docker build -t IMAGENAME d:\my-project
```

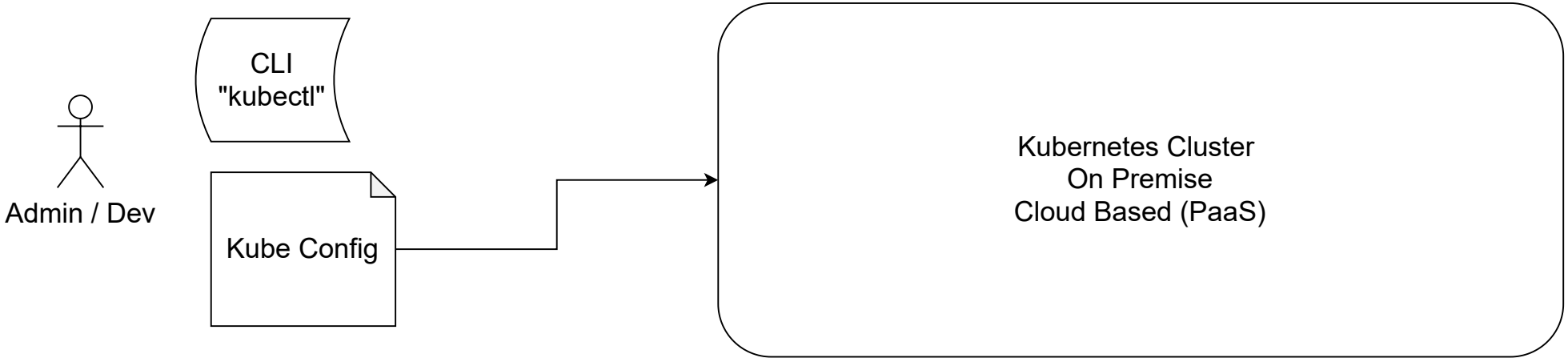
```
cd d:\my-project\
docker build -t IMAGENAME .
```

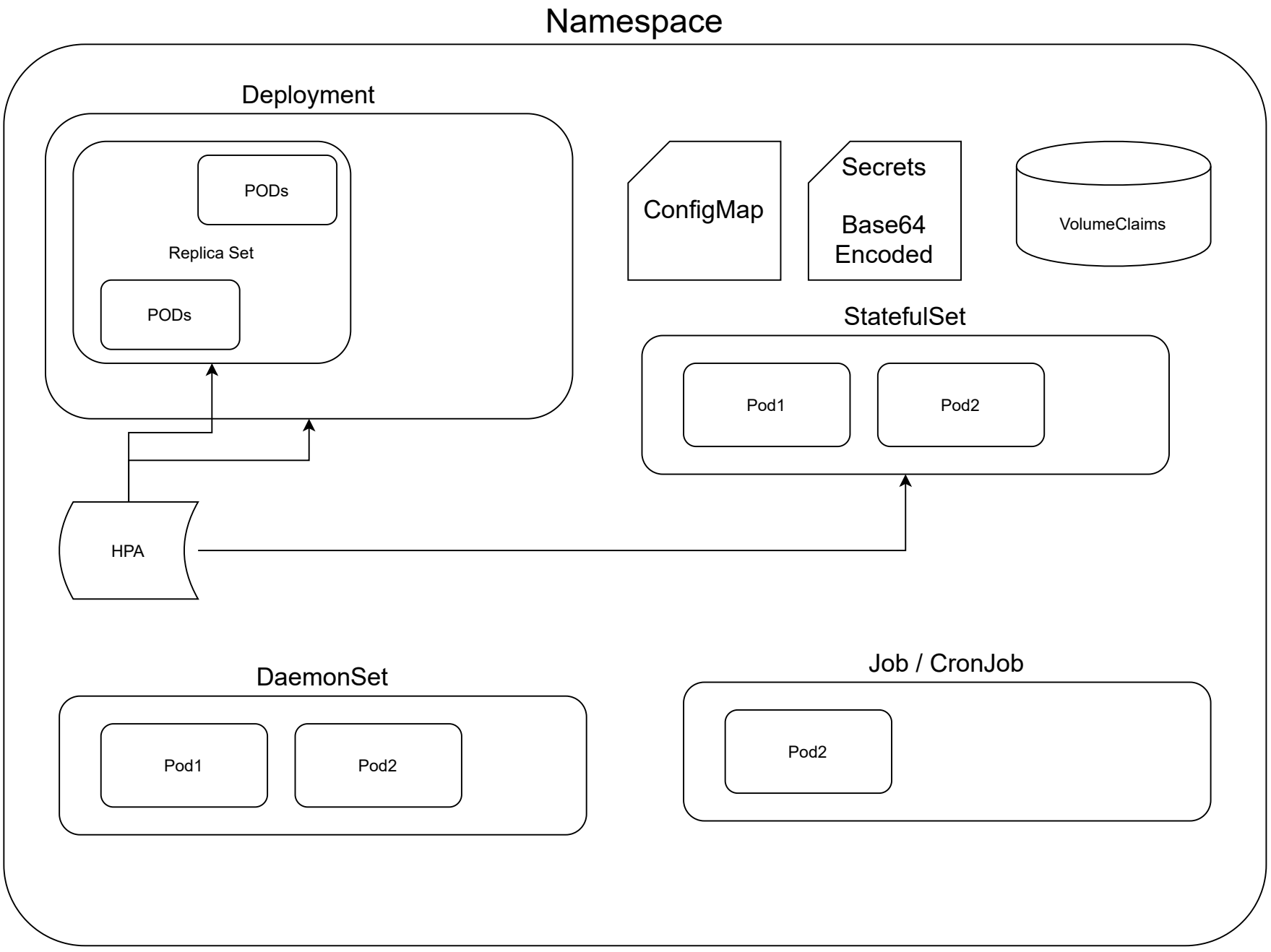
ADD app1.tar /usr/share/nginx/html/
===>
1. Copy the app1.tar inside container
2. Extract the contents of tar file
3. Delete the tar file


Kubernetes Cluster with HA
Master 1 (Leader)
Master 2 (follower)
Master 3 (follower)
Worker 4
Worker 8
Worker 3
Worker 7
Worker 2
Worker 6
Worker 1
Worker 5
Single Node Cluster
Docker Desktop , Minikube, MicroK8S
Master + Worker
AKS
System Workers
User Workers
System Workers
User Workers
System Workers
User Workers
Components Managed by Microsoft
ETC Data Store
API Server
Scheduler


EKS
AWS Managed Components
EC2 Instances
with Scheduler, API Server, ETCD, Controllers
....
EC2 As Worker
EC2 As Worker
EC2 As Worker
Fargate as Worker
Fargate as Worker
Fargate as Worker


Control Plane
Controller Manager
ETCD DataStore
Scheduler
API Server
kube-proxy
kubelet
Container Runtime
Master Nodes
kube-proxy
kubelet
Container Runtime
Worker Node


CLI
"kubectl"
Admin / Dev
Kube Config
Kubernetes Cluster
On Premise
Cloud Based (PaaS)


Namespace
Deployment
PODs
Replica Set
PODs
ConfigMap
Secrets
Base64
Encoded
VolumeClaims
StatefulSet
Pod1
Pod2
HPA
DaemonSet
Pod1
Pod2
Job / CronJob
Pod2