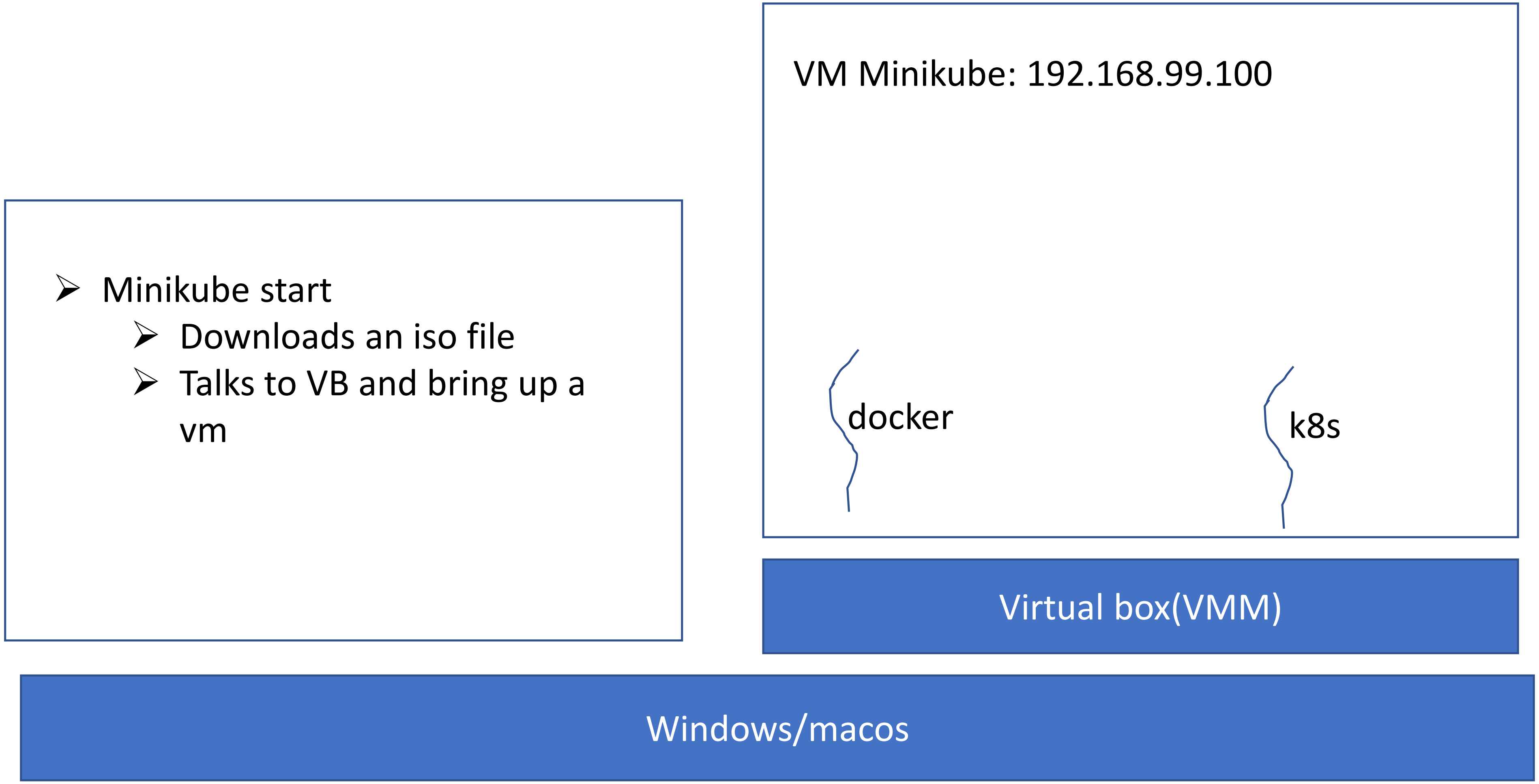
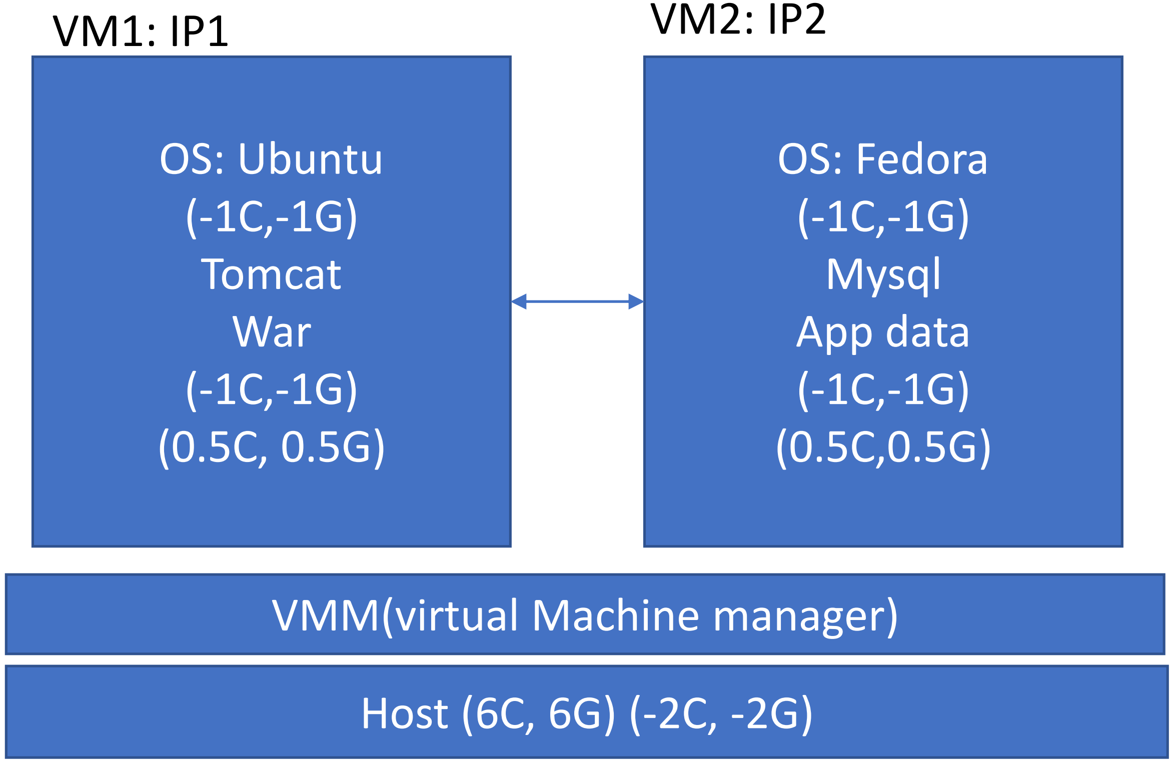


- Minikube start
    - Downloads an iso file
    - Talks to VB and bring up a vm

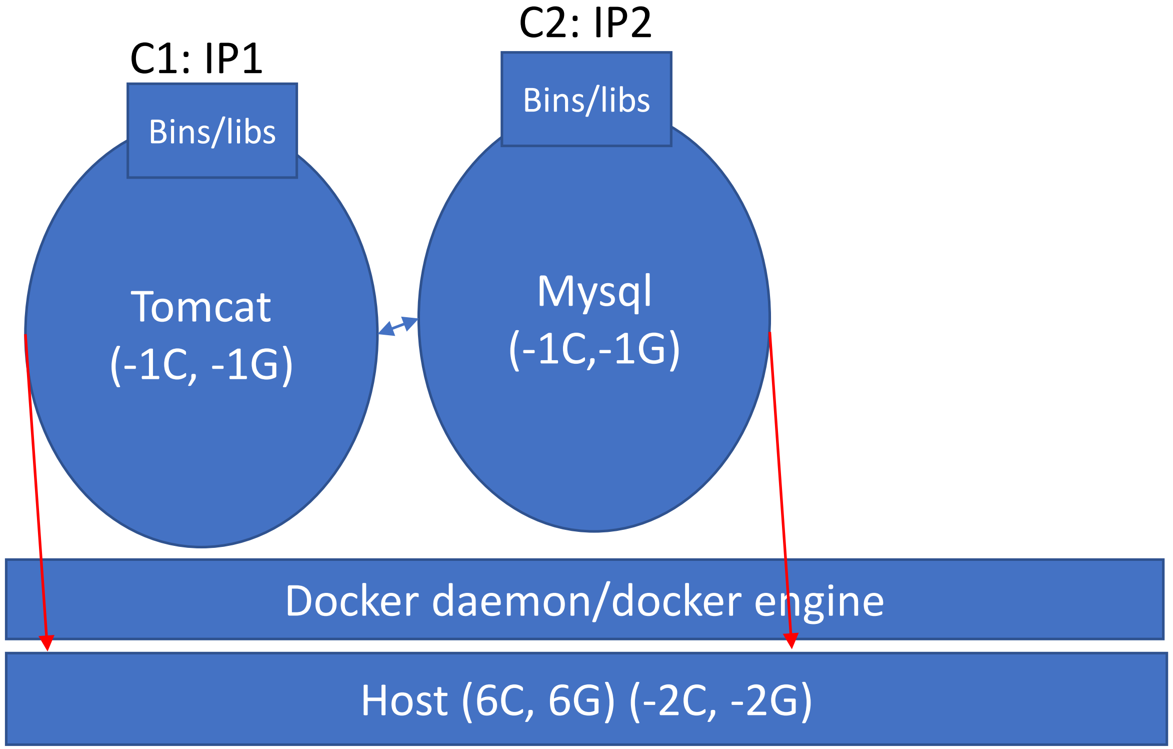
VM Minikube: 192.168.99.100

1. speed: mins
2. Share-ability: snapshot/checkpoint/appliance: 1.6 GB
3. Repeat-ability: run sheets, scripts
4. Bang-for-buck: limited, fragmented

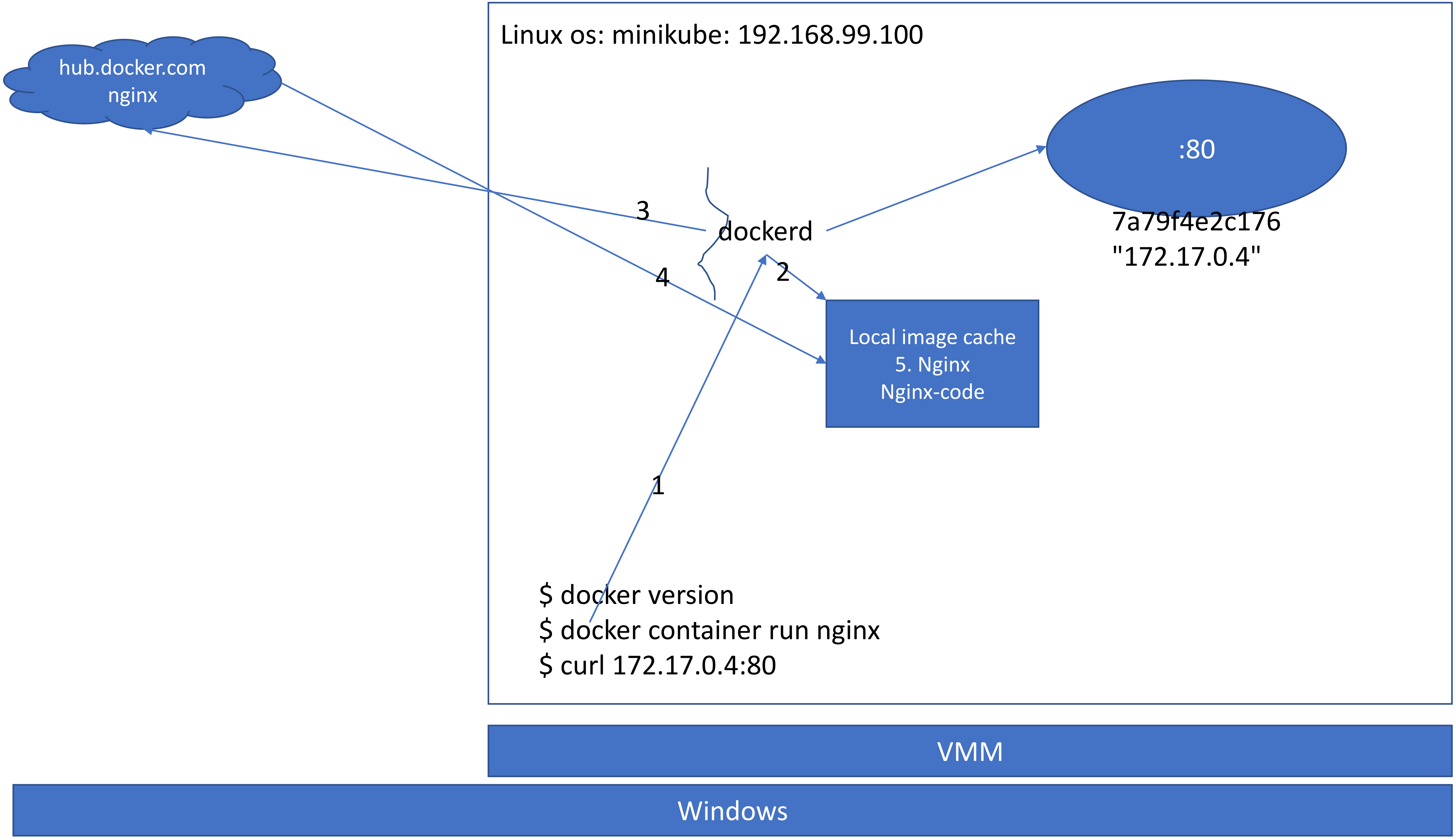


Isolation: Network, file system, process

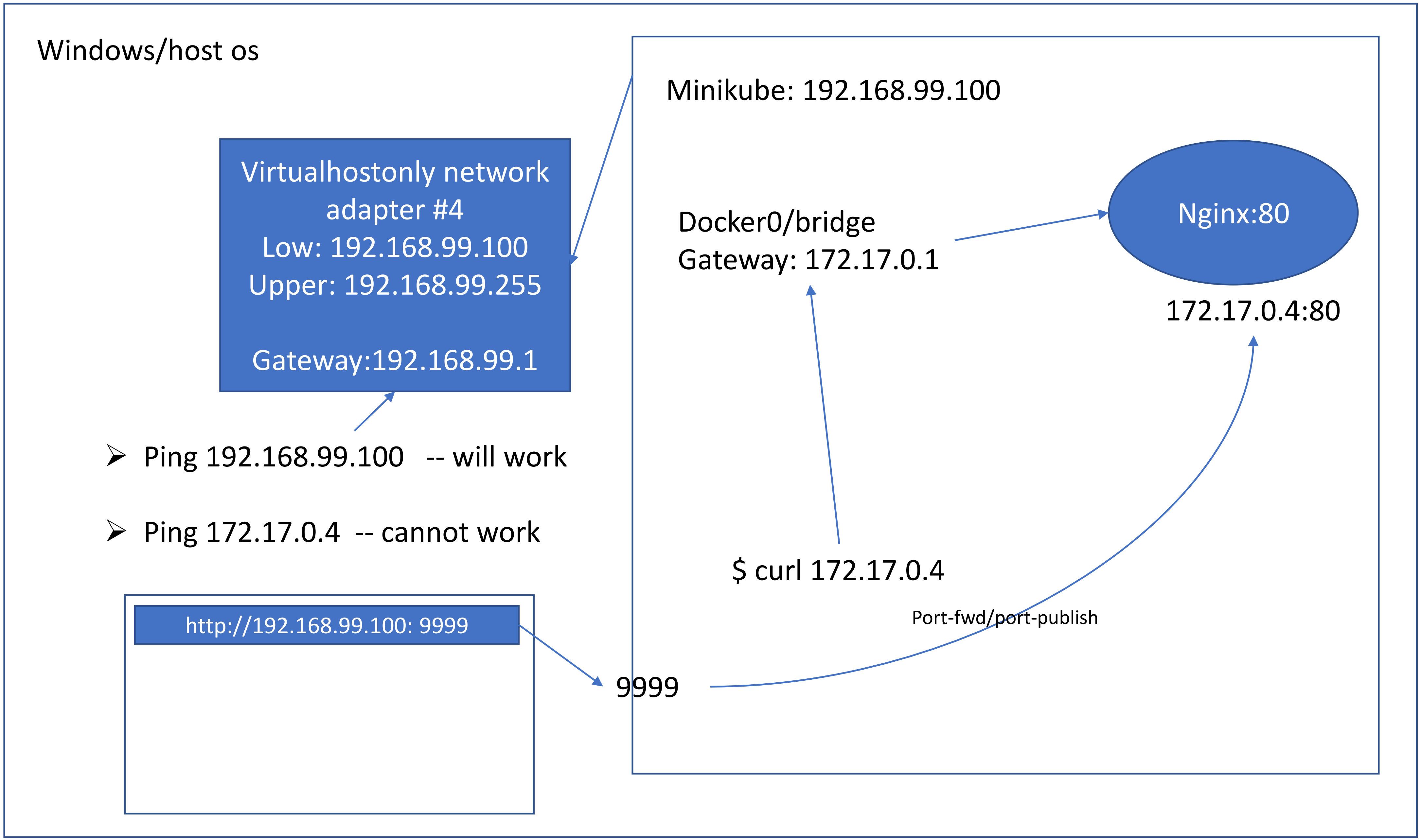
1. speed: ms
2. Share-ability: images
3. Repeat-ability: DSL, code
4. Bang-for-buck: ++, not fragmented



Isolation: Network, file system, process



```
$ docker version
$ docker container run nginx
$ curl 172.17.0.4:80
```



Windows/host os

Virtualhostonly network adapter #4
Low: 192.168.99.100
Upper: 192.168.99.255

Gateway:192.168.99.1

- Ping 192.168.99.100 -- will work
- Ping 172.17.0.4 -- cannot work

http://192.168.99.100: 9999

Minikube: 192.168.99.100

Docker0/bridge
Gateway: 172.17.0.1

Nginx:80

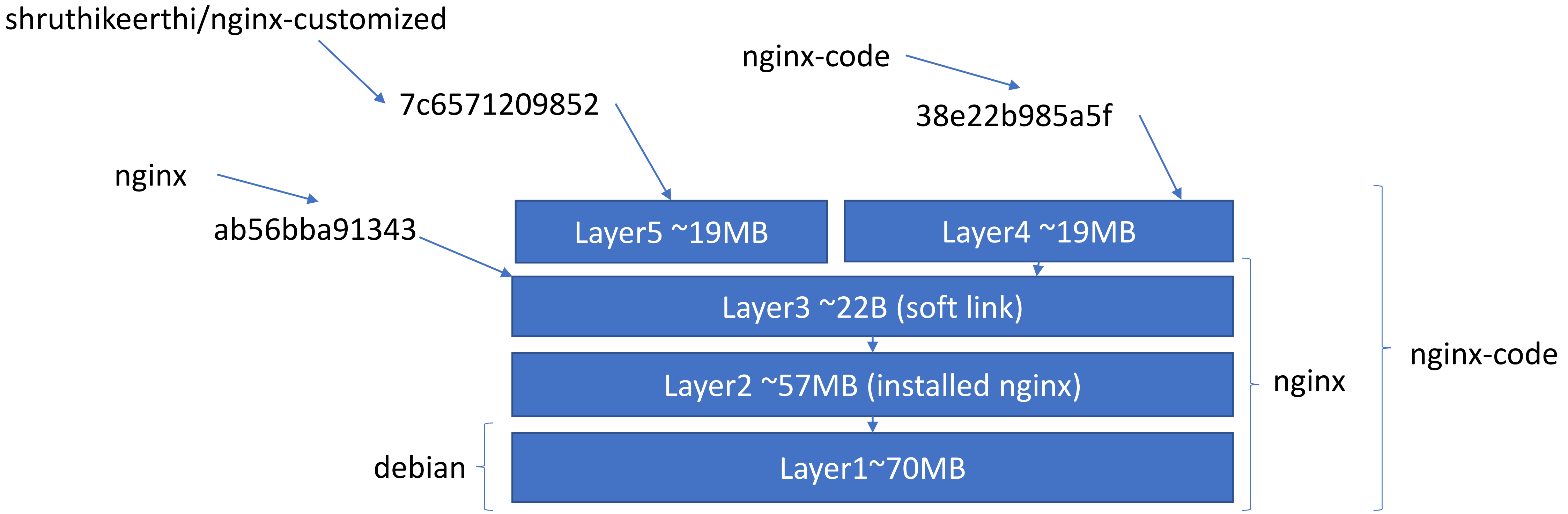
172.17.0.4:80

$ curl 172.17.0.4

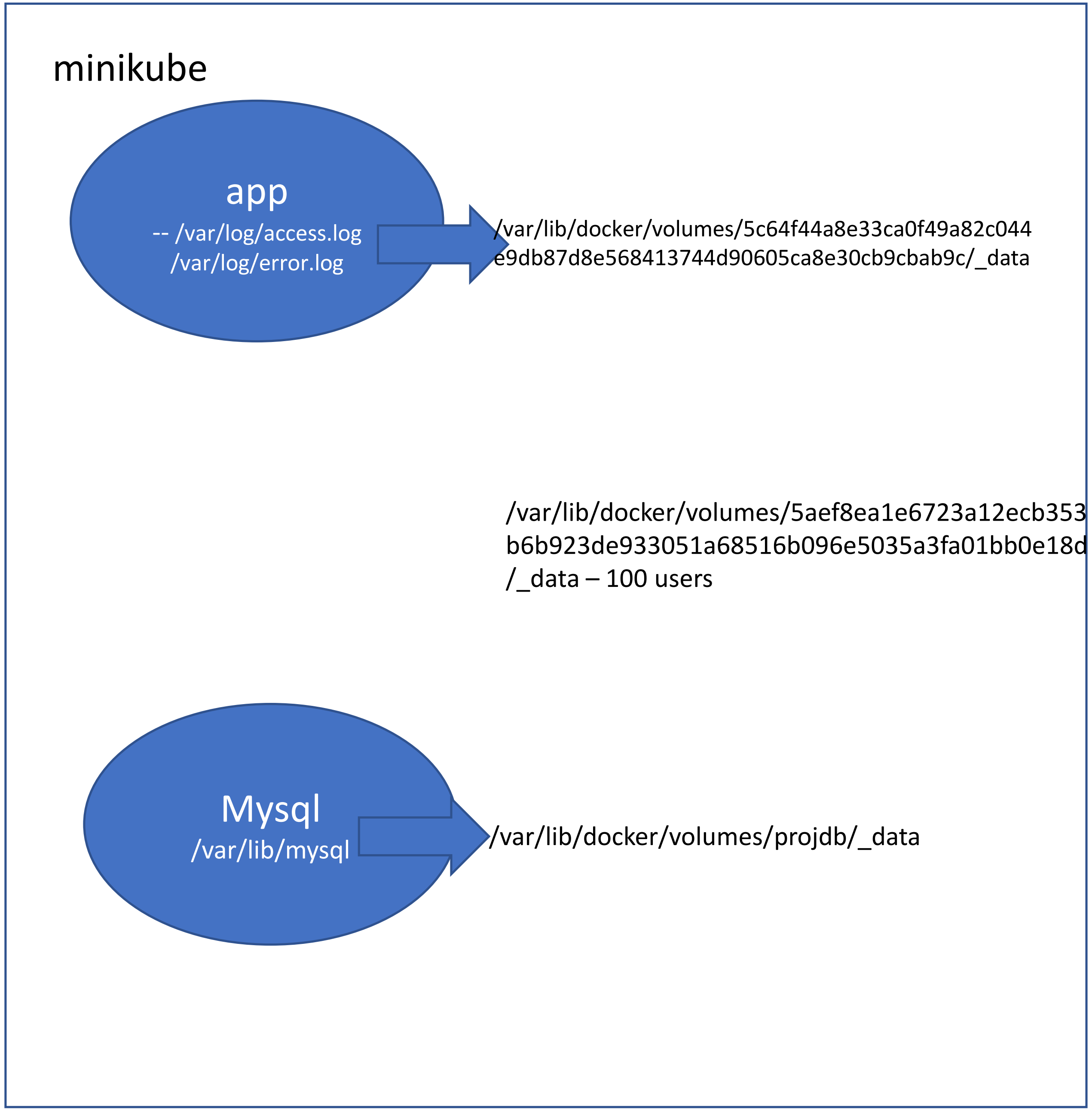
Port-fwd/port-publish

9999

Image: Unified File System



Containers fS : ephimeral



minikube

app

-- /var/log/access.log
/var/log/error.log

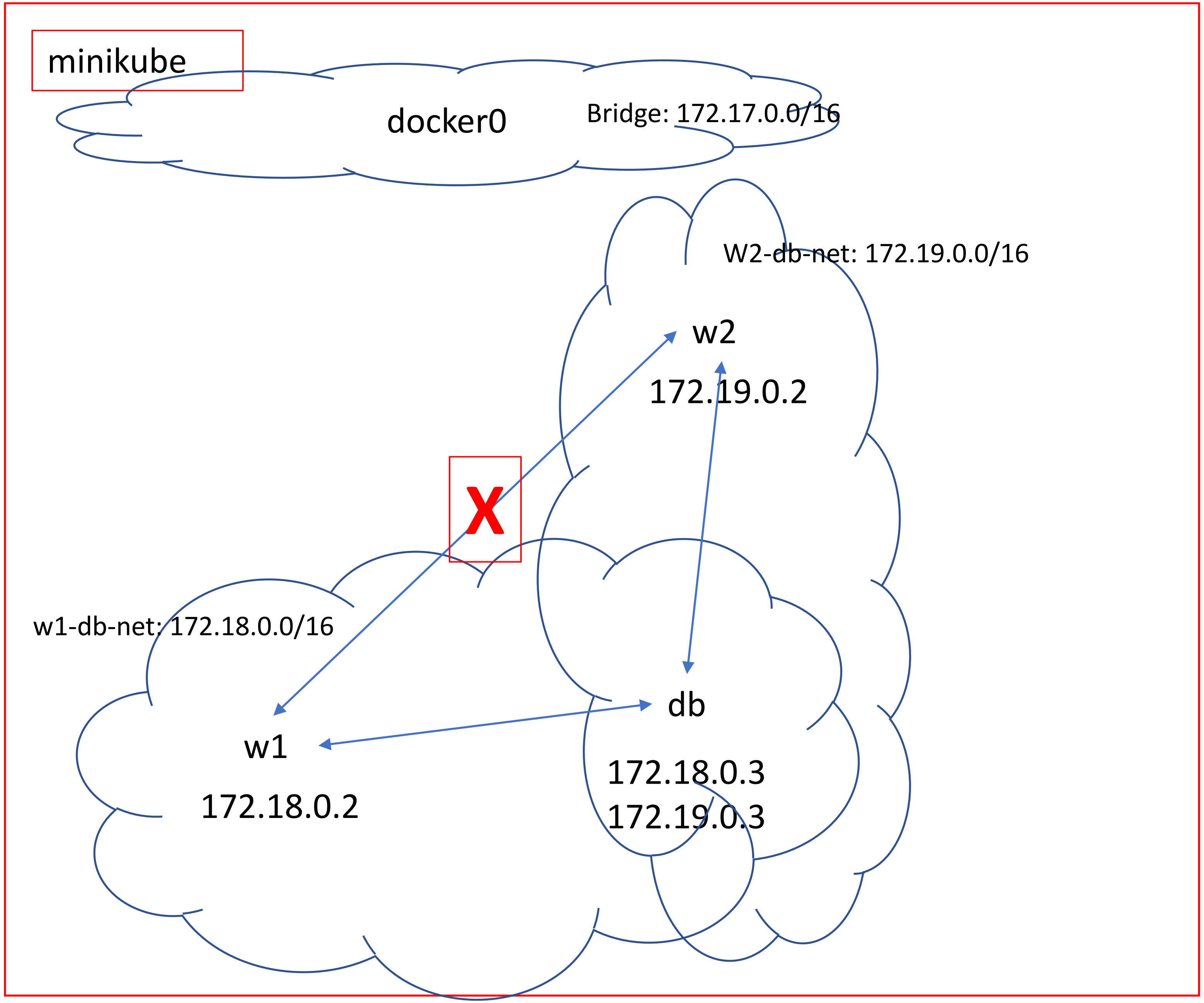
/var/lib/docker/volumes/5c64f44a8e33ca0f49a82c044
e9db87d8e568413744d90605ca8e30cb9cbab9c/_data

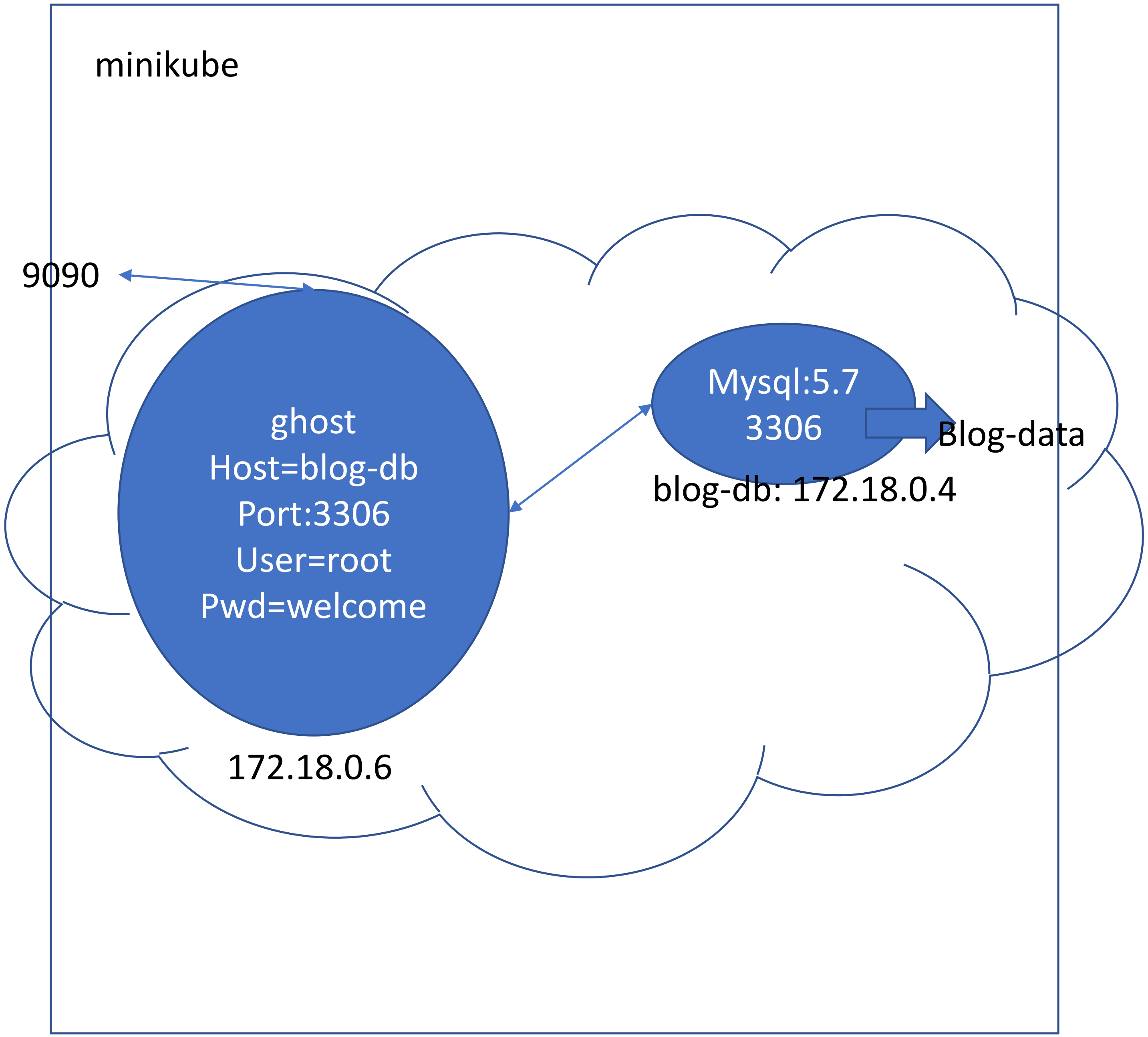
/var/lib/docker/volumes/5aef8ea1e6723a12ecb353
b6b923de933051a68516b096e5035a3fa01bb0e18d
/_data – 100 users

Mysql
/var/lib/mysql

/var/lib/docker/volumes/projdb/_data

1. ipaddress→ dns service
2. Better isolation: w1 -> db , w2 -> db, w1 – X - w2

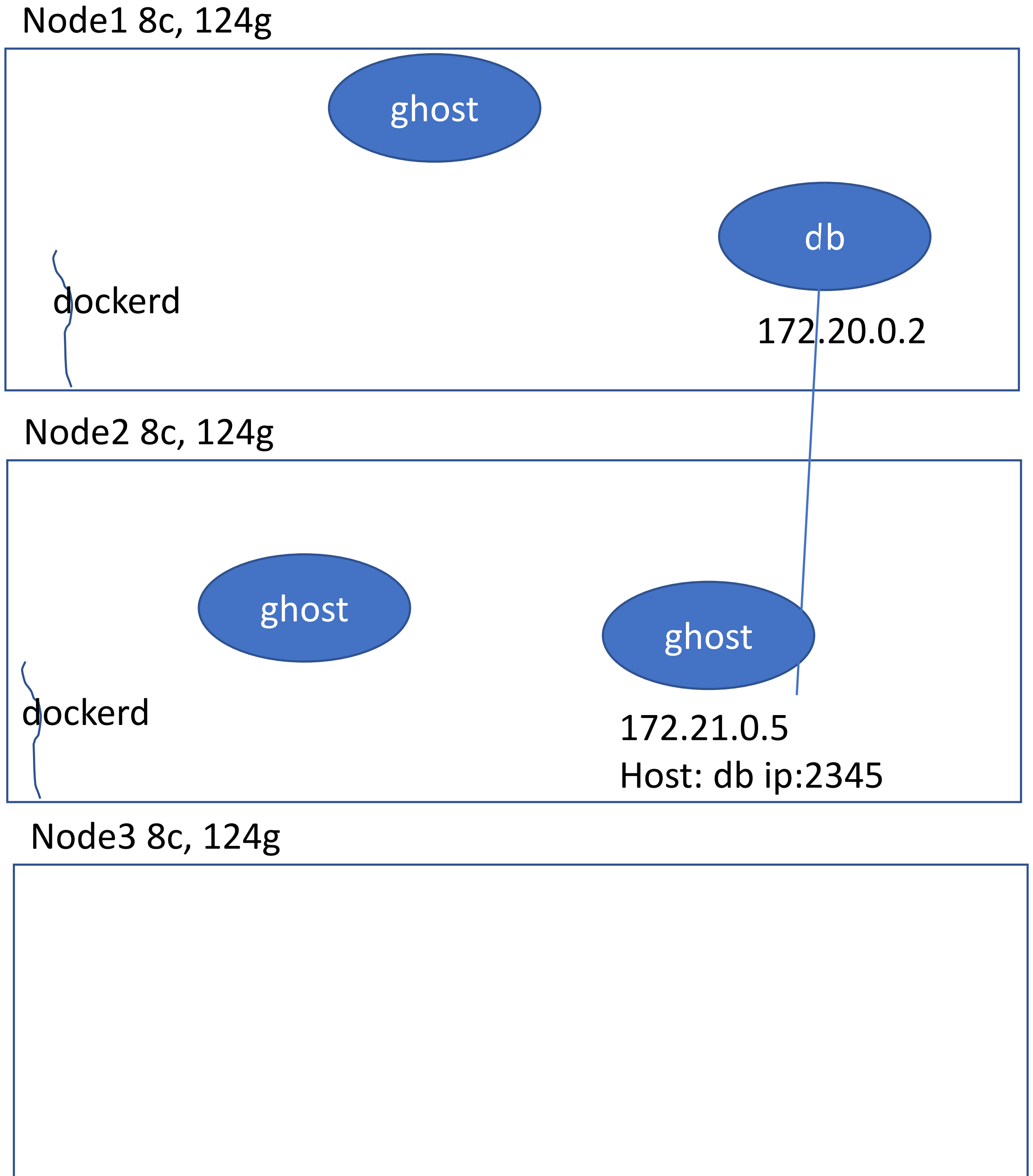



minikube
9090
ghost
Host=blog-db
Port:3306
User=root
Pwd=welcome
172.18.0.6
Mysql:5.7
3306
Blog-data
blog-db: 172.18.0.4

Container Orchestration

1. Application scaling
2. Load balancing
3. Interact and coordinate between multiple dockerd
4. Networking
   1. Ip address management
   2. No node awareness
5. Dns services
6. Node (VM, on prem,on cloud, physical) management -> cluster management

Products:
K8s
Docker swarm

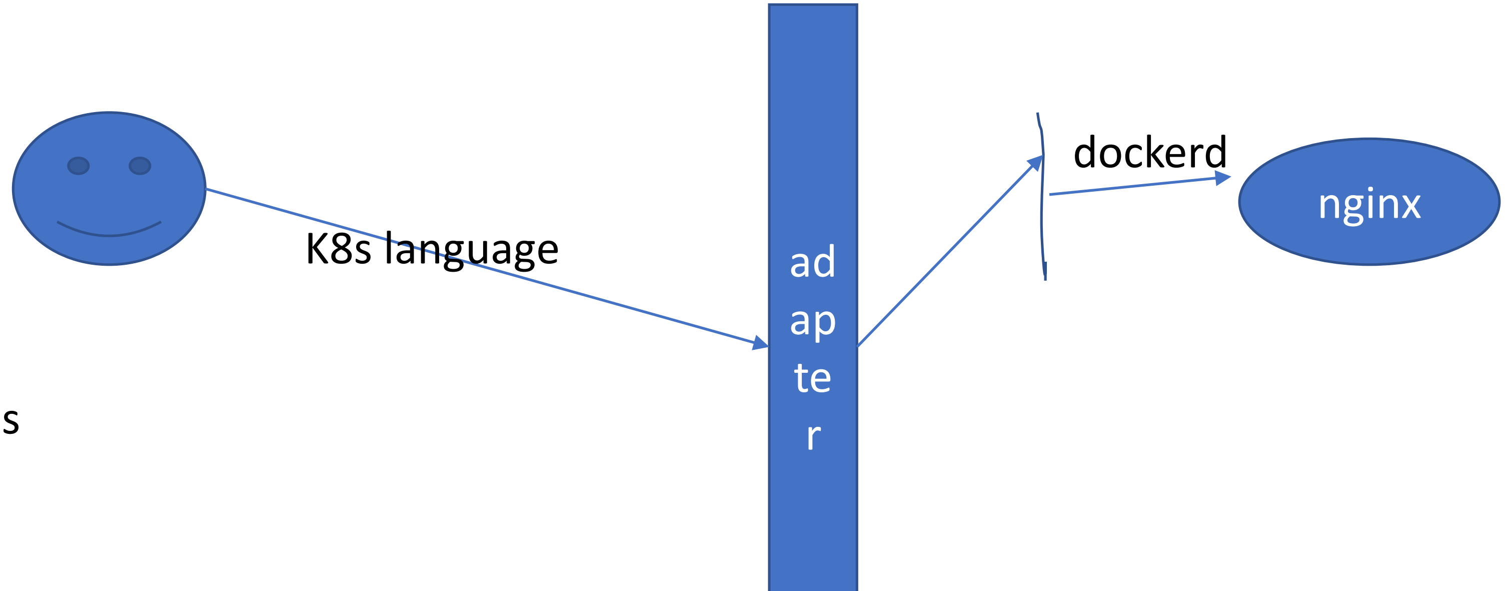


Docker Swarm

1. Open source, Docker
2. Native support for docker containers
3. Learning curve is smooth
4. DS → docker
5. Vendor-lockin
6. Fairly new

K8s

1. Open source, Google
2. Terminology or adapters
3. Learning curve is a litter steeper
4. K8s -> any container provider which implement Open Container Spec
5. No vendor lockin
6. Battle tested → benchmarked against 10000’s nodes



Docker – K8s

Learning K8s

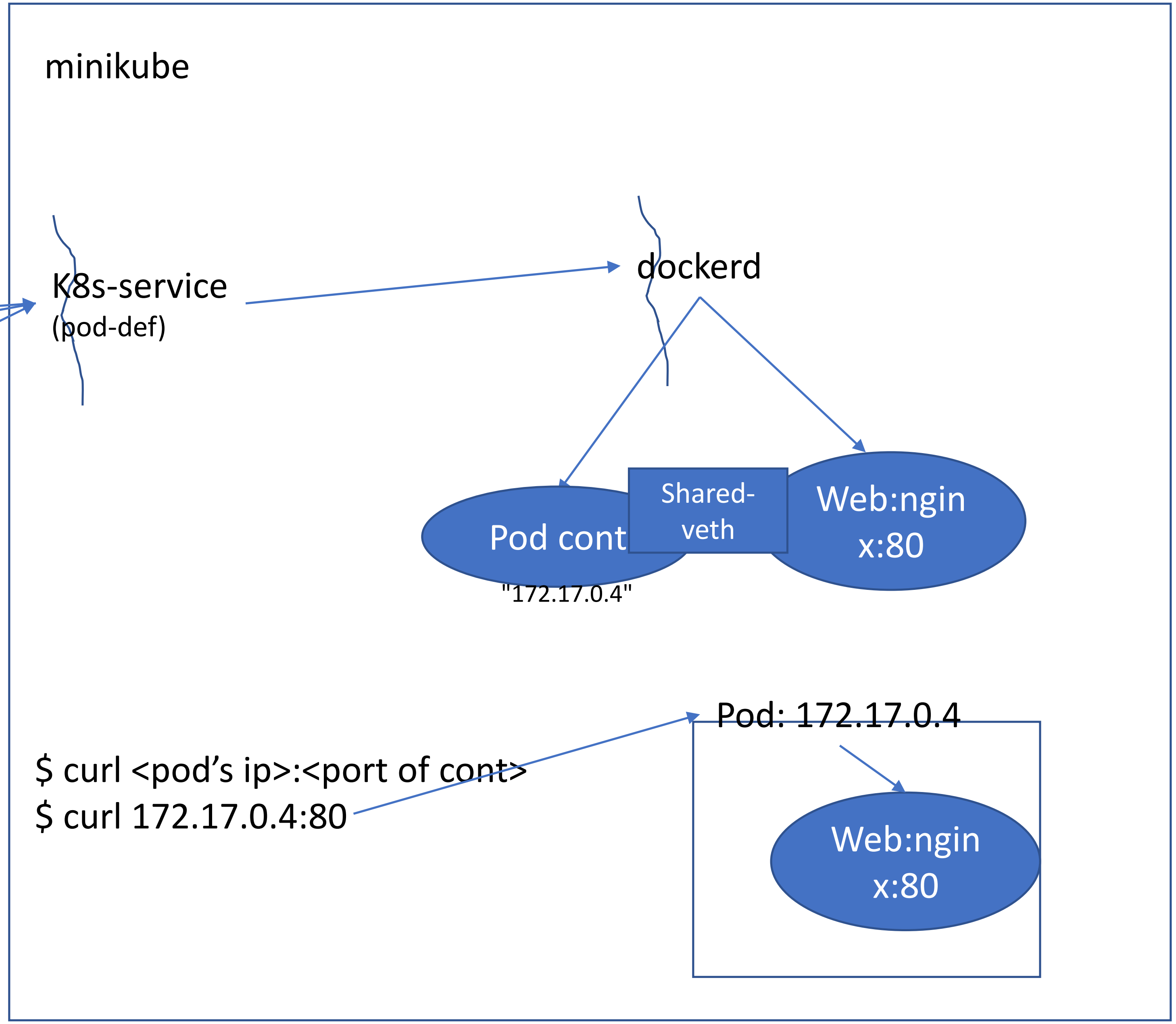
1. Terms and concepts k8s – single node cluster
2. Communication – single node clust
3. Architecture
4. Multi-node cluster k8s

- Kubectl (refers to ~/.kube/config)

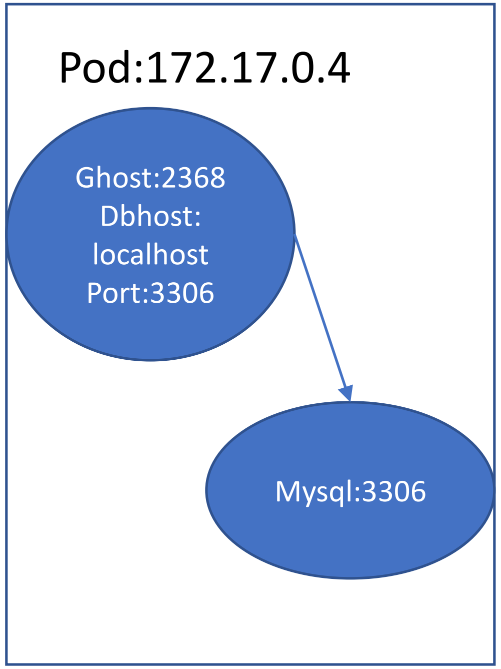
  Server version

- Kubectl apply –f pod.yml

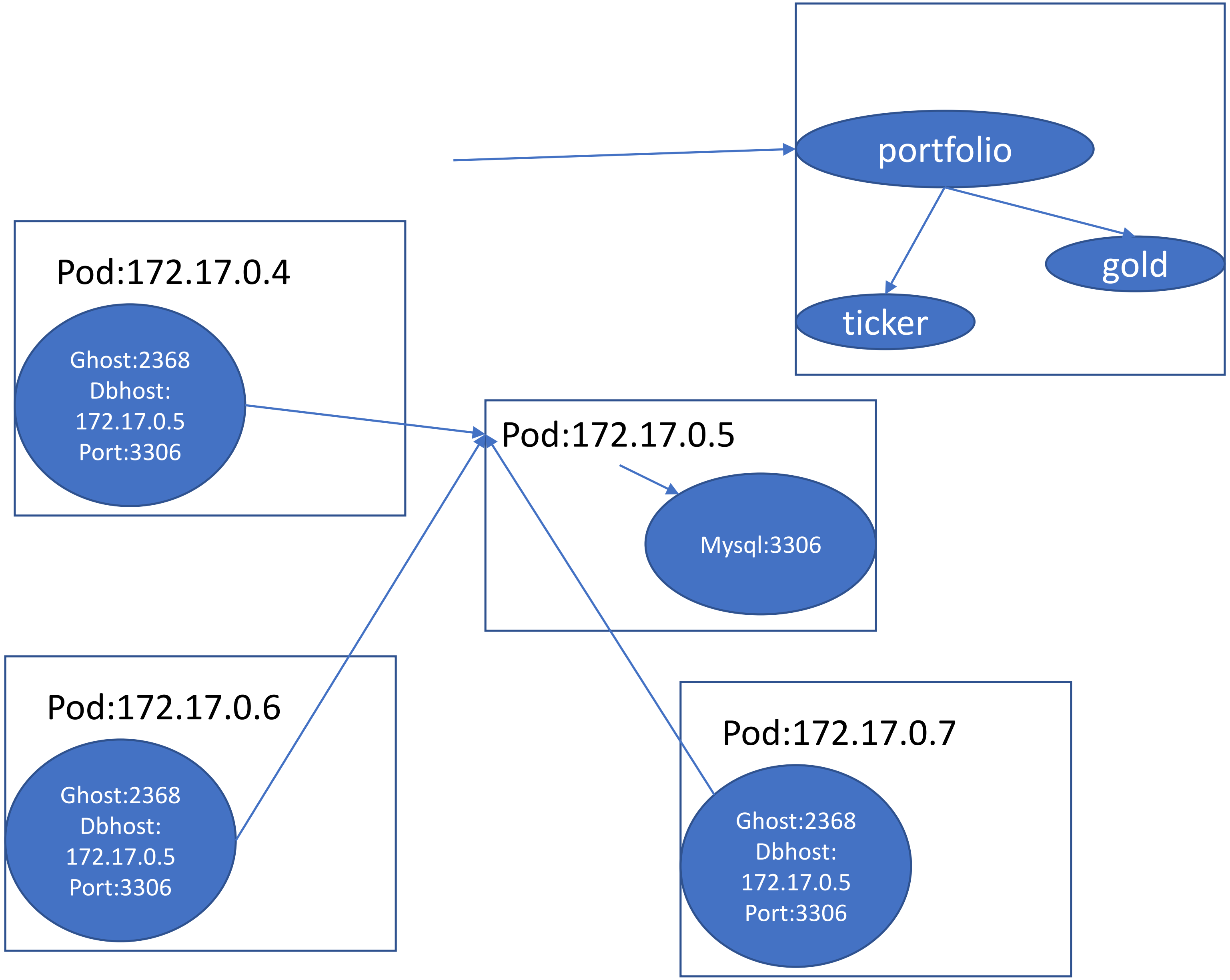
1. pod-cont provides n/w access to containers
2. Make sure that always the required containers run
3. Pod can have multiple containers provided they come up on different port number
4. Design time decision: Should I put containers in the same pod --- if they scalable together



```
$ curl <pod’s ip>:<port of cont>
$ curl 172.17.0.4:80
```



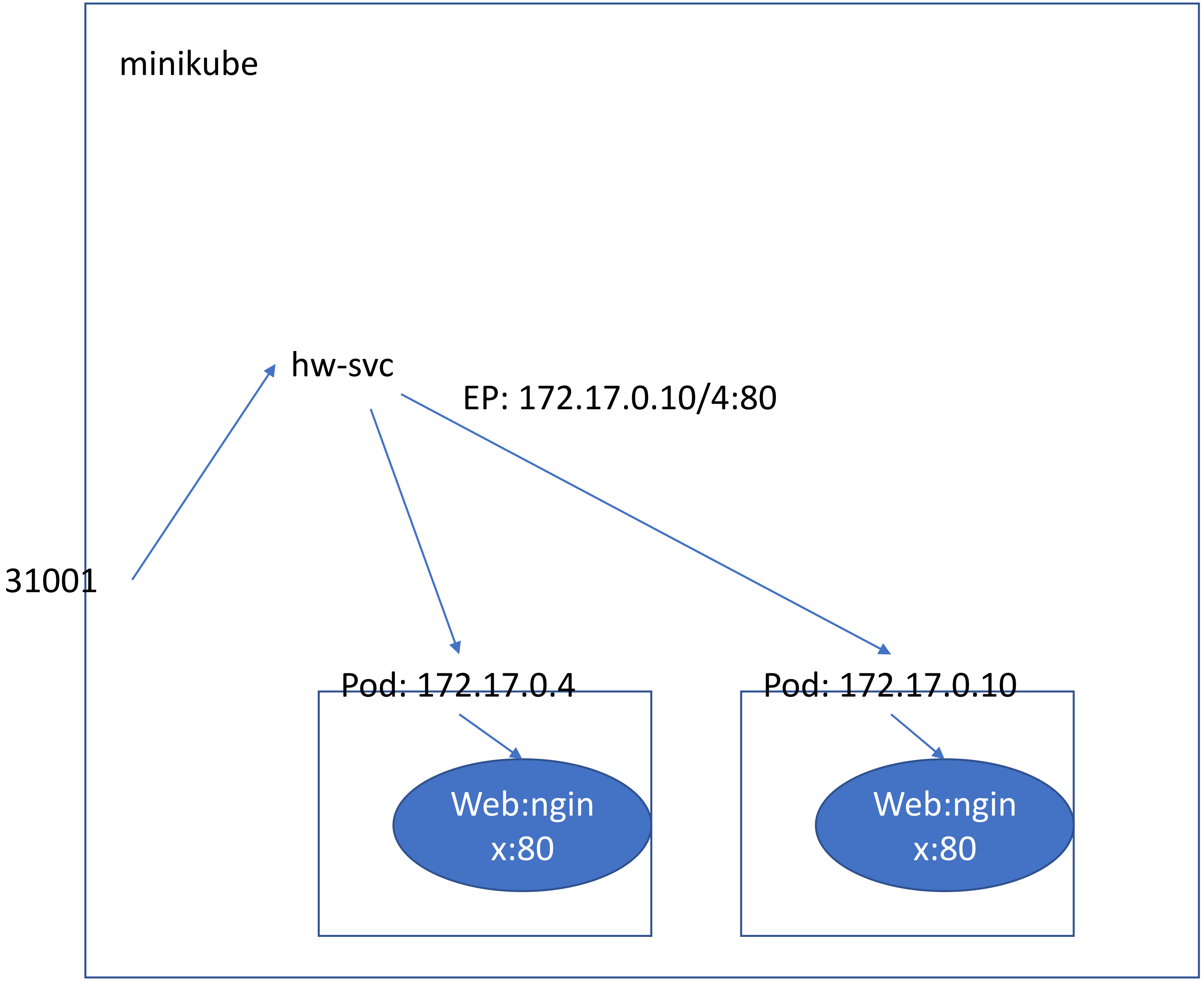
Deployment not scalable as it creates copies of DB too



Service

1. (opt): exposes a port on the node
2. Load balancing
3. Name mapping: the name of the svc itself will acts as a dns entry.

also called as Service Discovery



minikube

hw-svc

EP: 172.17.0.10/4:80

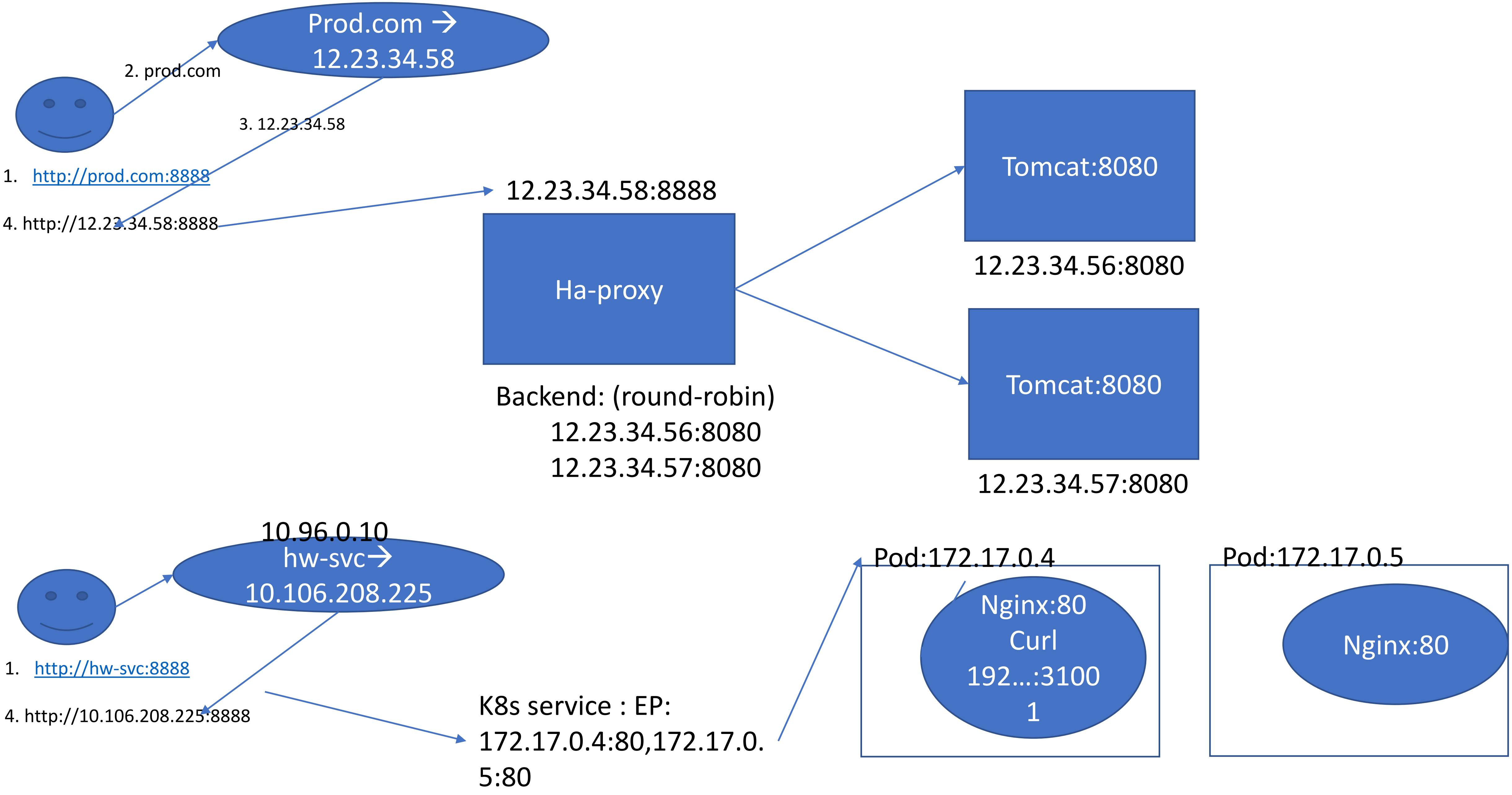
31001

Pod: 172.17.0.4

Pod: 172.17.0.10

Web:nginx:80

Web:nginx:80



Prod.com → 12.23.34.58

2. prod.com

3. 12.23.34.58

1. http://prod.com:8888

4. http://12.23.34.58:8888

12.23.34.58:8888

Ha-proxy

Backend: (round-robin)
12.23.34.56:8080
12.23.34.57:8080

Tomcat:8080

12.23.34.56:8080

Tomcat:8080

12.23.34.57:8080

10.96.0.10

hw-svc→ 10.106.208.225

1. http://hw-svc:8888

4. http://10.106.208.225:8888

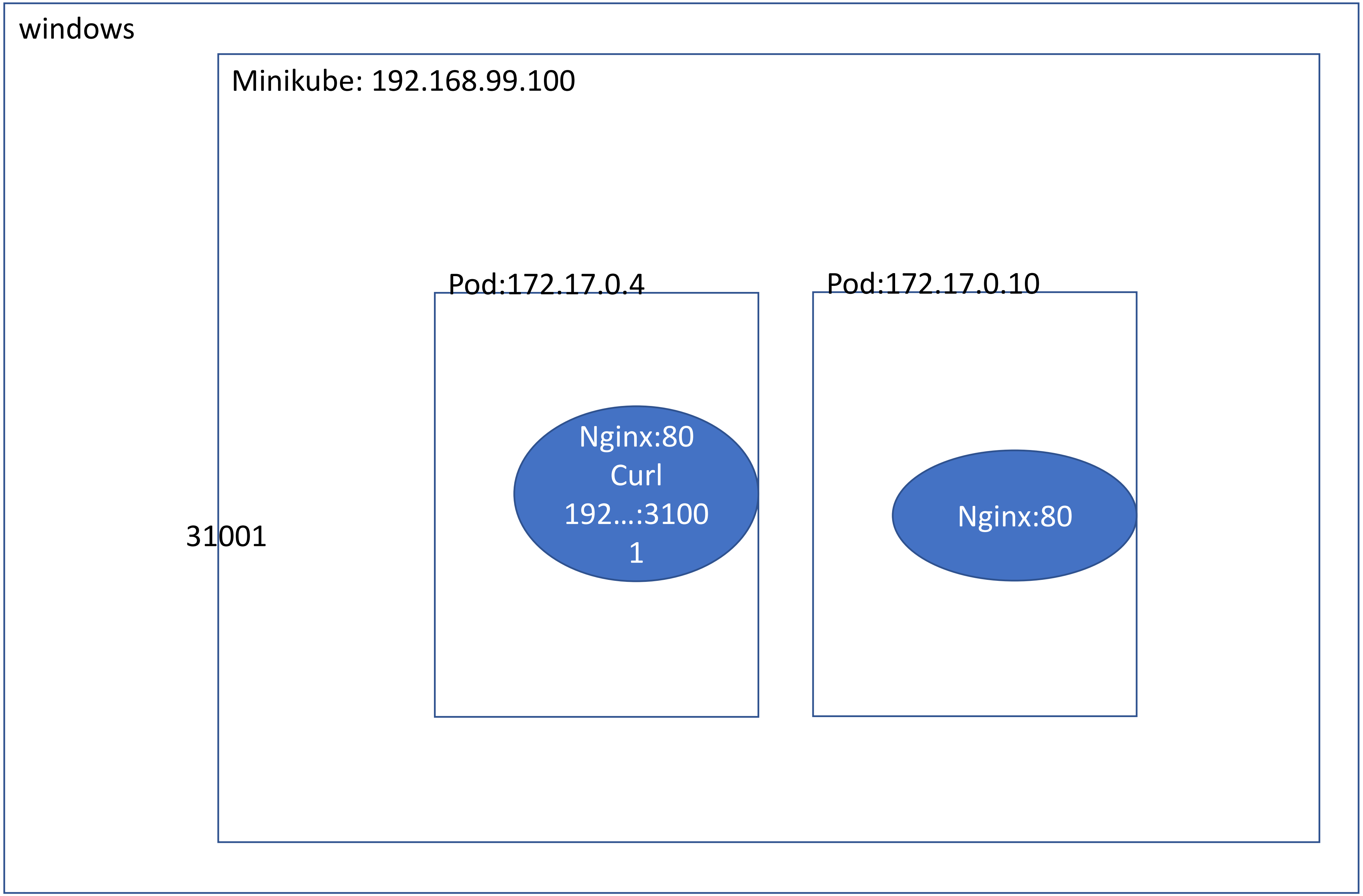
K8s service : EP:
172.17.0.4:80,172.17.0.5:80

Pod:172.17.0.4

Nginx:80
Curl
192…:3100
1

Pod:172.17.0.5

Nginx:80



windows

Minikube: 192.168.99.100

Pod:172.17.0.4

Nginx:80
Curl
192…:3100
1

Pod:172.17.0.10

Nginx:80

31001