www.infoq.com/cn

InfoQueue

@InfoQ

infoqchina

软件
正在改变世界！

Docker at Dianping

@kenshin54
http://github.com/kenshin54
i@kenshin54.me
jialin.tian@dianping.com

Agenda

- Introduction to Docker
- Docker at Dianping
- Why Container? Why Docker?
- Improvement
- Problems

Let’s start with

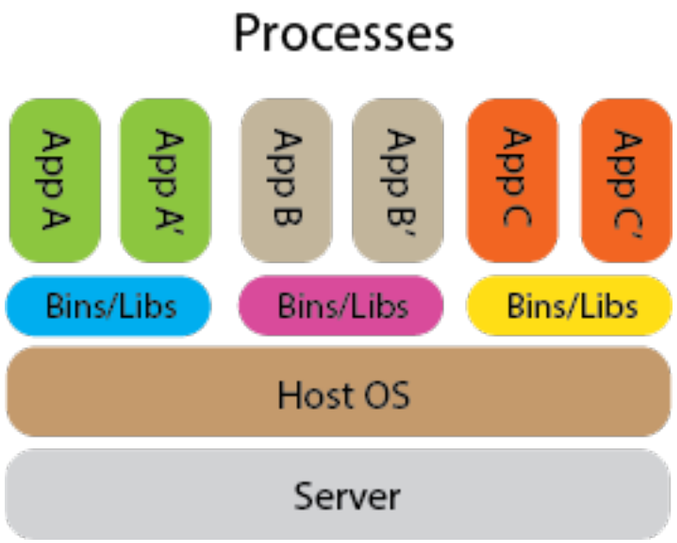
Questions

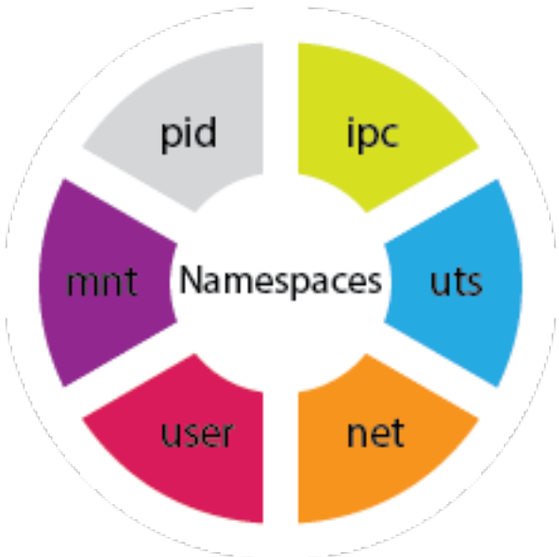
Raise your hand if you have…

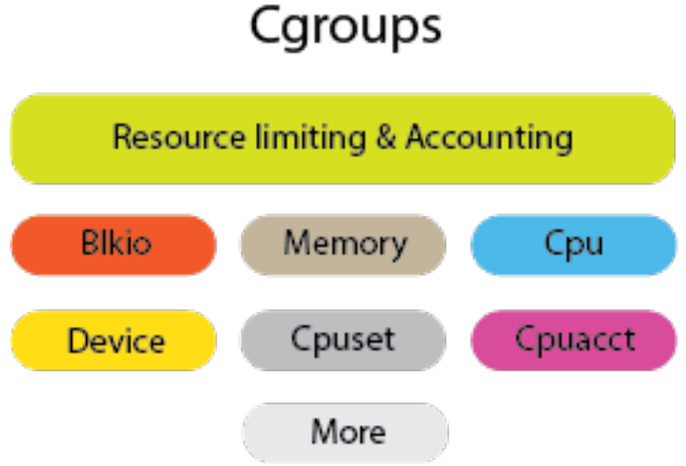
- Tried Docker (just for fun)
- Used Docker locally
- Deployed Docker for dev/test/production env

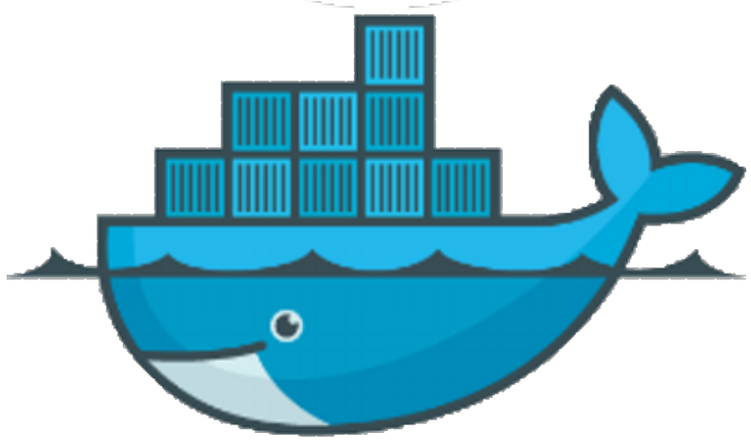
Docker Overview

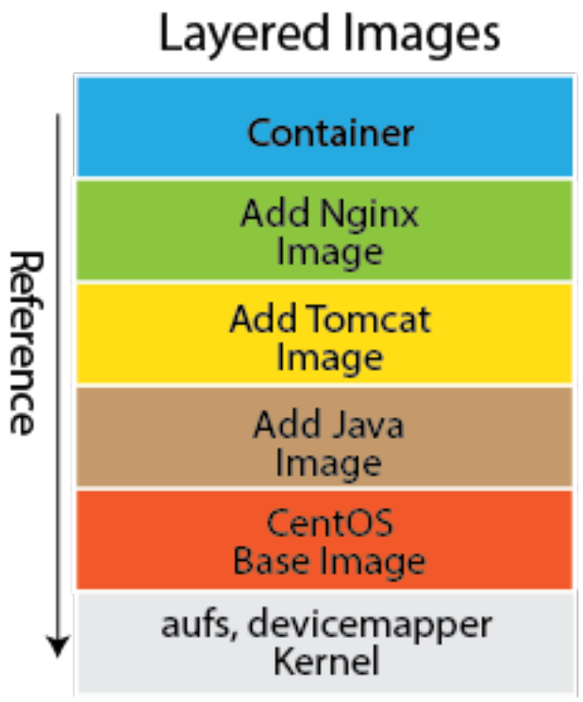
Build, Ship and Run Any App, Anywhere

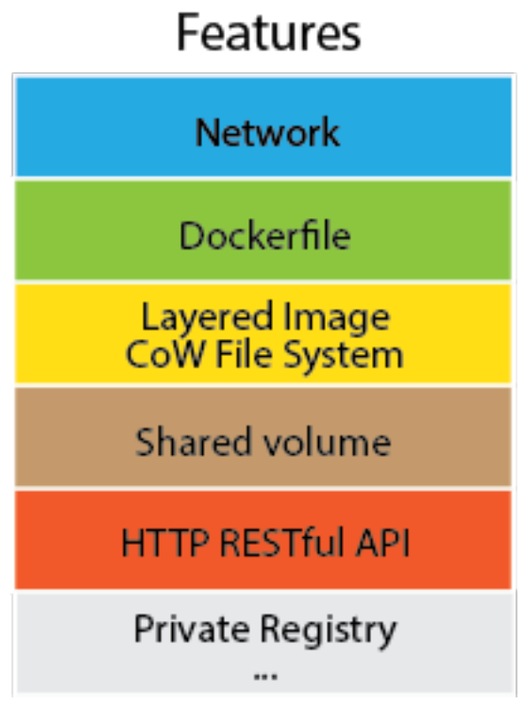












Docker at Dianping

- Why Container
- Why Docker
- Improvement
- Problems

Why Container

- We want to build a private cloud
- We want to improve resource utilization
- Application centric
- Easy to deploy and scale

Why Docker

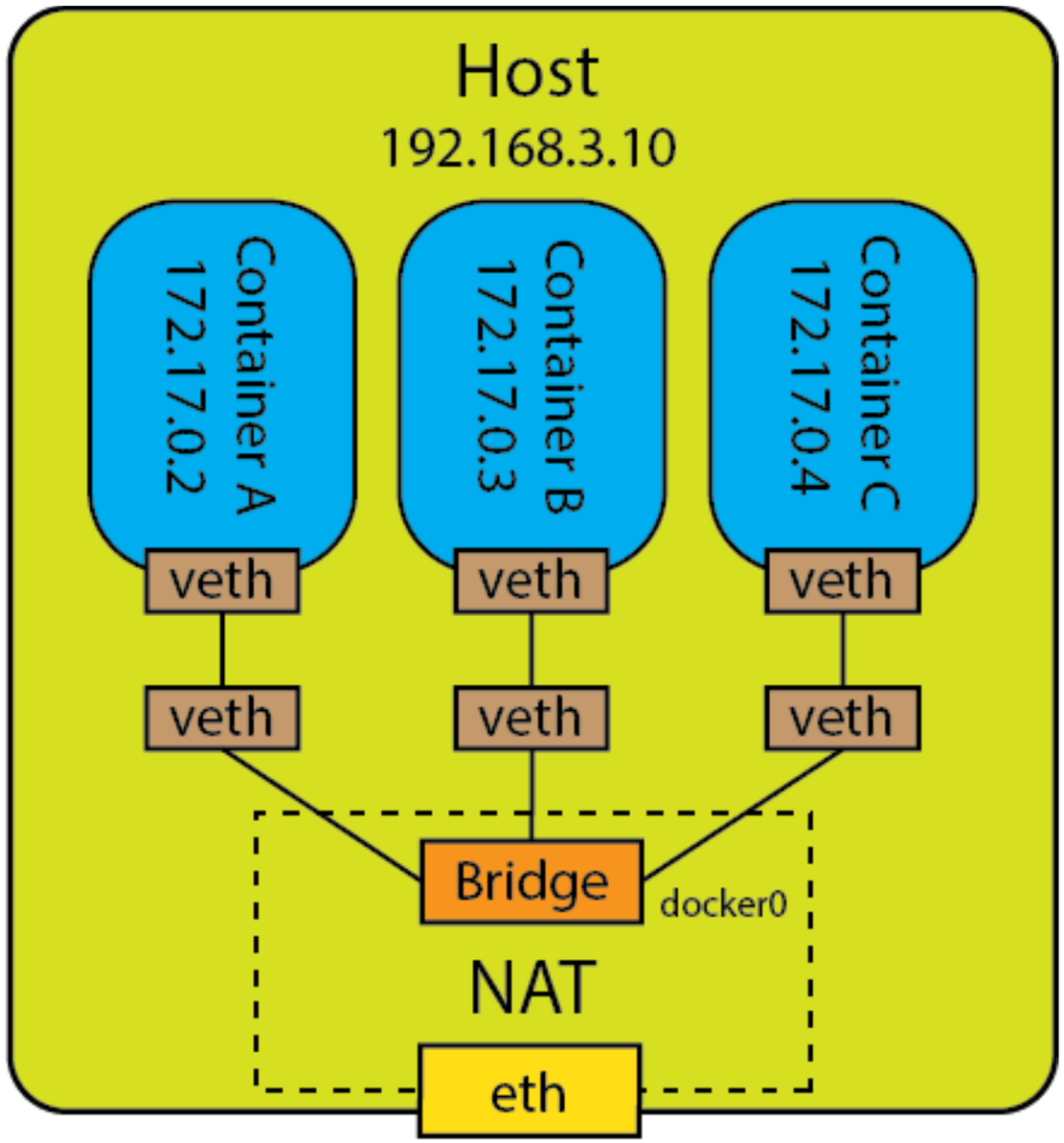
- Written in Go (LXC written in C)
  - GC, Concurrency…
- RESTful HTTP API
- Layered image
- Dockerfile (OPS & User friendly)
- Well-formed and detailed documentation
- Active communities and ecosystems

We need more …

- Seamless migration (DEV & OPS)
- Treat the container just like vm
- Public network
- Container metric collection/monitor
- Make system tools cgroup aware
- Execute command in running containers
- Change resources on the fly

We need more …

Network

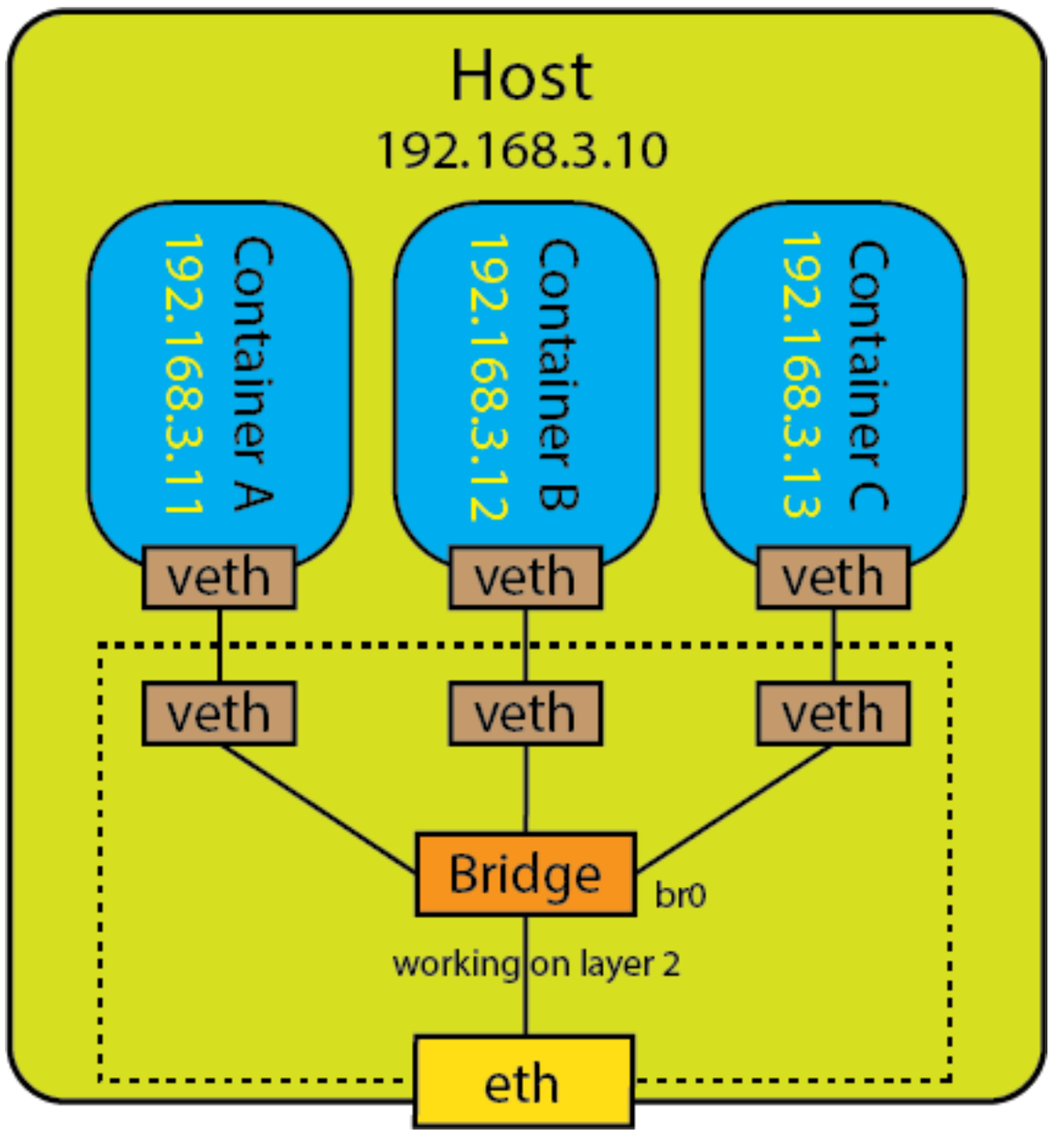


- Docker use host-only network with NAT by default
- Duplicated IP addresses in multiple hosts
  - Monitor unfriendly
- NAT is hard to maintenance
  - Service can not access directly (ssh…)

Solution (Open Source)

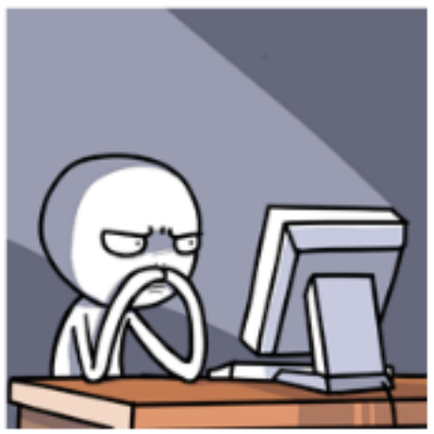
- Pipework Software-Defined Networking tools for LXC (LinuX Containers)
- Weave Weave creates a virtual network that connects Docker containers deployed across multiple hosts.
- Kubernetes Kubernetes is an open source implementation of container cluster management.

Solution



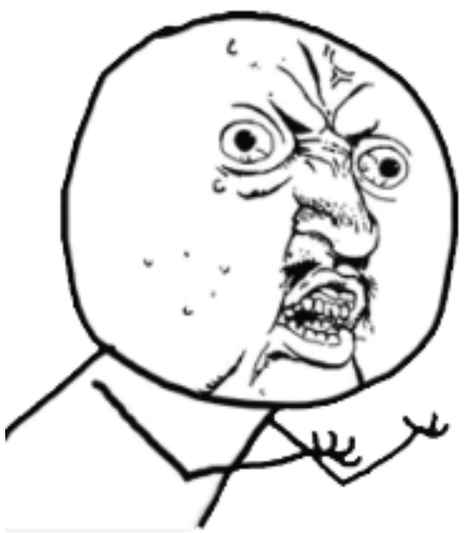
- Create a network bridge with a physical interface
- Specify network settings when creating containers
- docker run —ip=“192.168.3.11/24@192.168.3.1” tomcat
- Persistence support
- All services running on container are accessible directly

Tools & Metrics



I want to collect the stats of my server!

free
top
vmstat
iostat
sar
...



```
[root@ffc72a376a5f /]# grep -i "model name" /proc/cpuinfo | wc -l
16
[root@ffc72a376a5f /]# vmstat 1 3
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 2  0  22148 19913640 1683152 17781876    0    0     0     4    0    0  2  2 96  0  0
 0  0  22148 19914048 1683156 17781952    0    0     0    72 20349 32213  5  3 92  0  0
 0  0  22148 19915344 1683160 17782040    0    0     0   228 21194 33870  6  4 91  0  0
[root@ffc72a376a5f /]#
[root@ffc72a376a5f /]# free -m
             total       used       free     shared    buffers     cached
Mem:         64392      44946      19445          0       1643      17365
-/+ buffers/cache:      25937      38454
Swap:        65499         21      65478
[root@ffc72a376a5f /]#
```

Same as host!

Solution

- We collect metric data from cgroup
- docker metric container

```
root@test-01:~# docker metric 6b8780c60144
{"cpu_stats":{"cpu_usage":{"percent_usage":10.38961038961039,"current_usage":0.231419053795
67415,"total_usage":896785386411598,"percpu_usage":[0,0,0,0,0,0,0,0,228596394673195,2189053
87558297,223554353215896,225729250964210,0,0,0,0],"usage_in_kernelmode":284778270000000,"us
age_in_usermode":529647760000000},"throlling_data":{}},"memory_stats":{"usage":3823202304,"
max_usage":3833593856,"stats":{"active_anon":2204237824,"active_file":240865280,"cache":106
7270144,"hierarchical_memory_limit":3833593856,"hierarchical_memsw_limit":7667187712,"inact
ive_anon":551464960,"inactive_file":826404864,"mapped_file":10244096,"pgfault":365020825,"p
gmajfault":1080,"pgpgin":137358355,"pgpgout":136425012,"rss":2755702784,"rss_huge":0,"swap"
:4493312,"total_active_anon":2204237824,"total_active_file":240865280,"total_cache":1067270
144,"total_inactive_anon":551464960,"total_inactive_file":826404864,"total_mapped_file":102
44096,"total_pgfault":365020825,"total_pgmajfault":1080,"total_pgpgin":137358355,"total_pgp
gout":136425012,"total_rss":2755702784,"total_rss_huge":0,"total_swap":4493312,"total_unevi
ctable":0,"unevictable":0},"failcnt":1106371},"blkio_stats":{},"freezer_stats":{}}
```

Kernel Patch

- Apply Alibaba's kernel patch to CentOS 6.5

- Make the /proc VFS cgroup aware

Exec Command
in running container

- Upgrade app without restart container
- Easy to attach into container (bash…)

Solution

- We extend docker with the exec feature
- docker exec container command [args…]

```
root@test-01:~# docker exec 6b8780c60144 ps -e
  PID TTY          TIME CMD
    1 ?        00:00:00 start.sh
   26 ?        00:12:30 supervisord
   28 ?        00:00:00 sshd
   57 ?        00:00:17 crond
  132 ?        10-08:47:24 java
42588 ?        00:00:00 ps
root@test-01:~# docker exec 6b8780c60144 ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
96: eth0: <BROADCAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 56:48:6b:9b:c1:d3 brd ff:ff:ff:ff:ff:ff
```

Resource Control

- Docker can set resource while creating a container, but didn’t provide a way to change resource when the container is running

Solution

- Allow to change resource on the fly
- docker cgroup container memory.limit_in_bytes=2G

```
root@test-01:~# docker cgroup 8acbc71d344c memory.limit_in_bytes cpuset.cpus
[{"Subsystem":"memory.limit_in_bytes","Out":"2147483648","Err":"","Status":0},{"Subsys
tem":"cpuset.cpus","Out":"0-1","Err":"","Status":0}]

root@test-01:~# docker cgroup 8acbc71d344c memory.limit_in_bytes=1g cpuset.cpus=2-3
[{"Subsystem":"memory.limit_in_bytes","Out":"1g","Err":"","Status":0},{"Subsystem":"cp
uset.cpus","Out":"2-3","Err":"","Status":0}]

root@test-01:~# docker cgroup 8acbc71d344c memory.limit_in_bytes cpuset.cpus
[{"Subsystem":"memory.limit_in_bytes","Out":"1073741824","Err":"","Status":0},{"Subsys
tem":"cpuset.cpus","Out":"2-3","Err":"","Status":0}]
```

Image…

- Build an app-agnostic base image
- Reduce the cost of image maintenance
- Bind the app with volume feature

Problems

- Remove container is slow
- Host inaccessible in public network
- “Storm traffic” in public network

Remove container is slow!

- We use device mapper as storage driver
- 10 seconds docker takes to remove a container
- Docker use sparse file as storage by default which need to do block discard manually

Solution

- Use a raw block device to improve performance

```
docker -d --storage-driver=devicemapper --storage-opt dm.basesize=40G
--storage-opt dm.datadev=/dev/sda2 ...
```

Host inaccessible

- After start/stop a container, the host is inaccessible in a short period

- By default bridge interfaces in Linux use, for their MAC address, the lowest MAC address among the enslaved interfaces.

Solution

- Config a fixed MAC address in network configuration

```
DEVICE="br0"
TYPE="Bridge"
BOOTPROTO=static
ONBOOT="yes"
IPADDR=192.168.3.17
NETMASK=255.255.255.0
MACADDR=A0:D3:C1:04:78:68
```

“Storm traffic”

- After shutdown a container
- Huge traffic would be caused in the same subnet
- Be accompanied with dropped packets

Why?

- Physical network interface enter promiscuous mode
- Docker enable ip_forward by default
- Container will inherit ip_forward configuration

Solution

- turn off ip_forward

- sysctl -w net.ipv4.ip_forward=0

Migration

- 500+ Apps
- 120+ Apps, 400+ Instances
- 60 -> 23 Hosts
- 3 seconds to scale

Summary

- Docker is a fantastic tool for distributed applications
- Build, Ship and Run Any App, Anywhere
- Less resource consumption
- Application scale painless
- Docker containers are like Legos, you can combine them in your ways

Q & A

Thank you