

Intermediate Docker (Kubernetes) ICP – K8S

Agile Brains Consulting, Inc.



Agile Brains Consulting

©Agile Brains Consulting Inc. All rights reserved. Not to be reproduced without prior written consent.



Introduction !!

- Who you are
- Your role in the team
- Experience with Docker and Kubernetes

Agenda

Chapter 1 – Docker Refresher

Chapter 2 – Microservices Patterns

Chapter 3 – Kubernetes Installation and Architecture

Chapter 4 – Kubernetes API overview and resources

Chapter 5 – Kubernetes for CI

Chapter 6 – Kubernetes Monitoring, Logging and health checks

Chapter 1: Docker – Refresher



Lab: Setting up the machine

- Install docker, Kubernetes in the provided AWS machine
- Configure Kubernetes master
- Verify installation

*** Master node need to have at least 2 CPUs*

https://github.com/shekhar2010us/kubernetes_teach_git/blob/master/kubernetes_single_node_cluster_installation.md

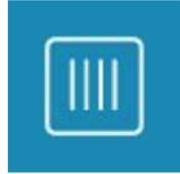


Docker Basics



Docker Image

- The basis of a Docker container



Docker Container

- The standard unit in which the application service resides



Docker Engine

- Creates, ships and runs Docker containers deployable on physical or virtual host locally, in a datacenter or cloud service provider

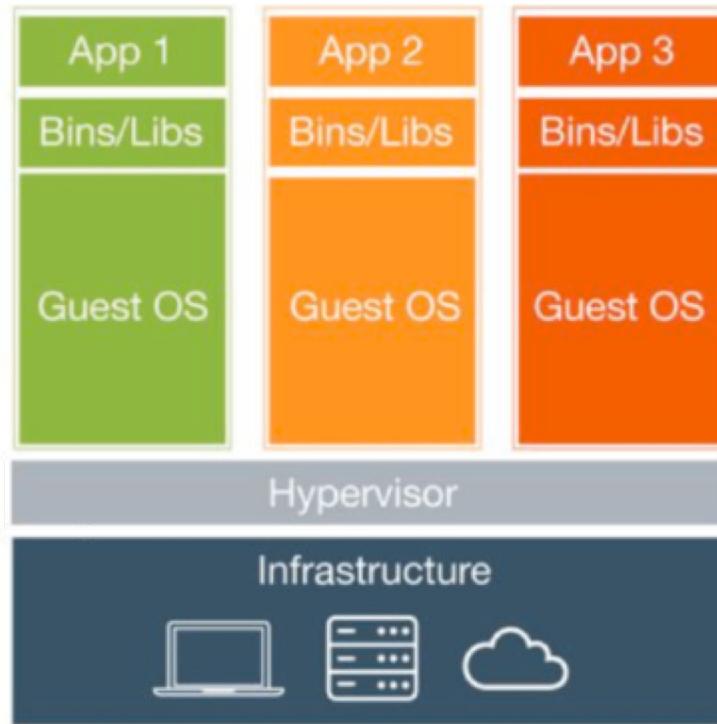


Docker Registry

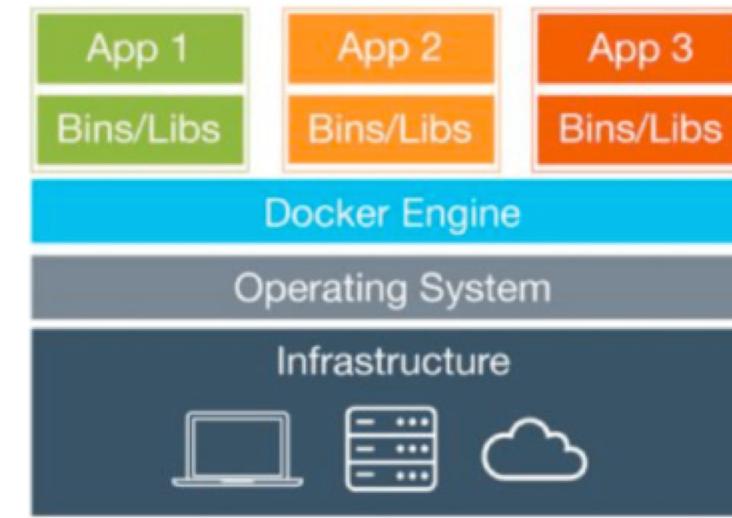
- On-premises registry for image storing and collaboration



VMs vs Containers



Virtual Machines



Containers



Containers as a Solution

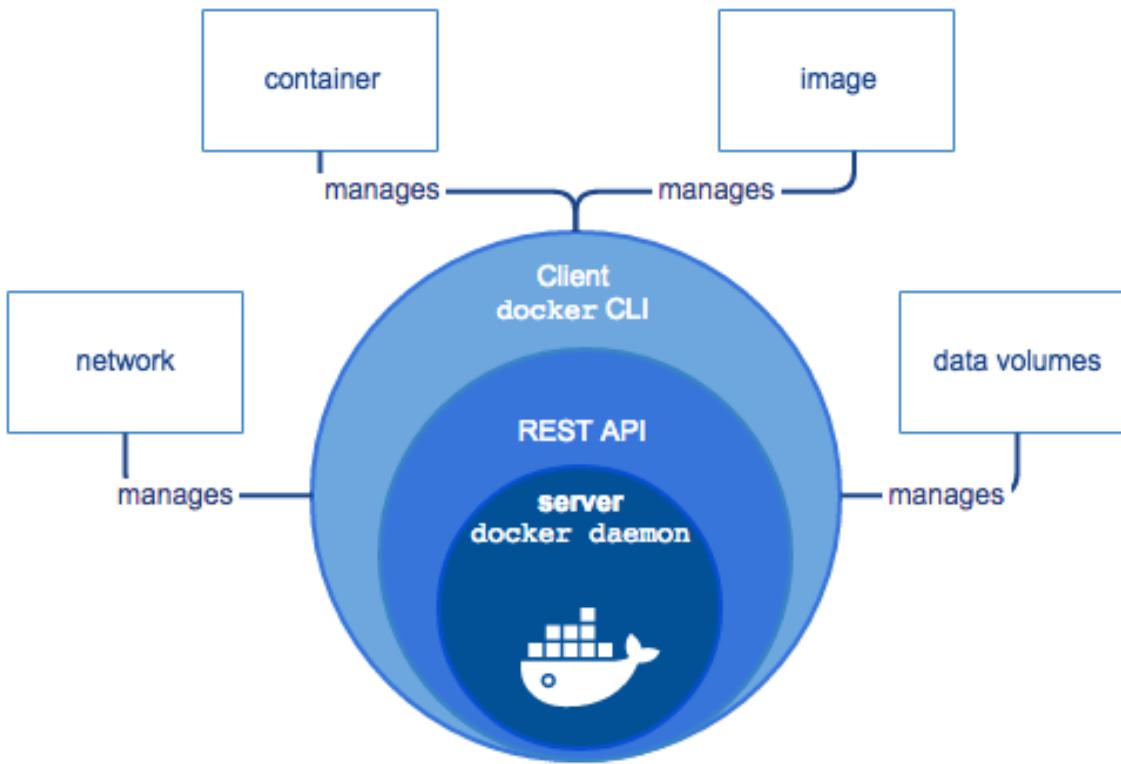


Container

- Packages up software binaries and dependencies
- Isolates software from each other
- Container is a standard format
- Easily portable across environment
- Allows ecosystem to develop around its standard



Components of Docker



Components of Docker

- The Docker daemon (**dockerd**) is a service runs on the host OS. It listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes.
- The Docker client (**docker**) is the primary way that many Docker users interact with Docker. When you use commands such as docker run, the client sends these commands to dockerd, which carries them out. The docker command uses the Docker API.
- A Docker **registry** stores Docker images. Docker Hub and Docker Cloud are public registries that anyone can use, and Docker is configured to look for images on Docker Hub by default. You can even run your own private registry.

Connect Client to remote dockerd

Server: Create CA private and public keys on server, create CSR, and generate signed certificate

Client: Create client key and make the key suitable for authentication

Connection: docker --tlsverify --tlscacert=ca.pem --tlscert=cert.pem --tlskey=key.pem \ -H=\$HOST:2376 version



Installing Docker on Linux

- There is a Docker-maintained community script that will install the latest release on your Linux machine (most popular flavors) available at <https://get.docker.com/>
- Run ‘curl -sSL https://get.docker.com/ | sh’ in your terminal
- The script will also install a Union File System driver and verify Docker engine functionality



Check Docker Installation

\$ sudo docker version

```
cludwig — student@docker-proto: ~ — ssh student@...  
student@docker-proto:~$ sudo docker version  
Client:  
Version: 17.03.1-ce  
API version: 1.27  
Go version: go1.7.5  
Git commit: c6d412e  
Built: Mon Mar 27 17:14:09 2017  
OS/Arch: linux/amd64  
  
Server:  
Version: 17.03.1-ce  
API version: 1.27 (minimum version 1.12)  
Go version: go1.7.5  
Git commit: c6d412e  
Built: Mon Mar 27 17:14:09 2017  
OS/Arch: linux/amd64  
Experimental: false  
student@docker-proto:~$
```

\$ sudo service docker status

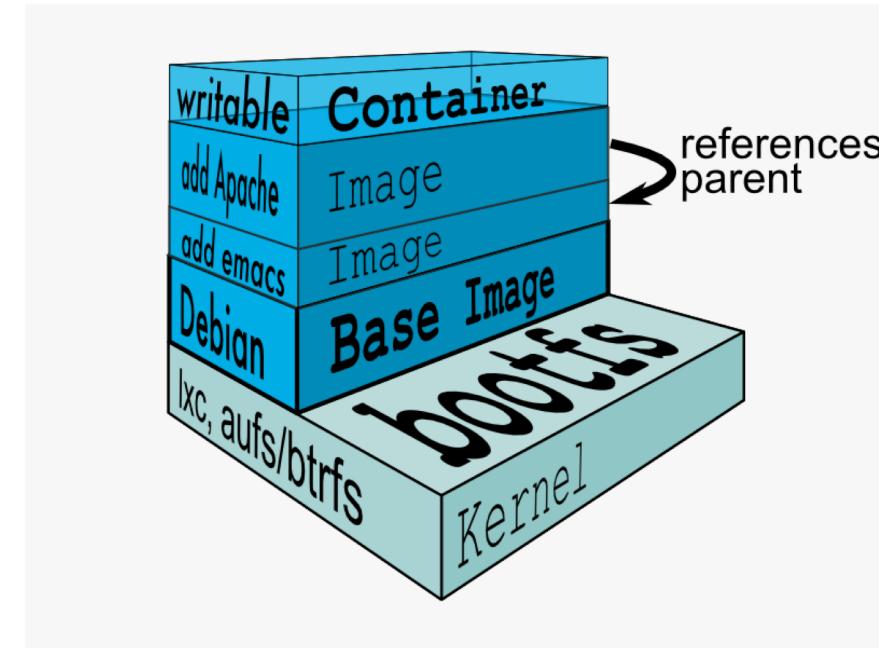
```
[ubuntu@ubuntu-xenial:~/dockers$ sudo service docker status  
● docker.service - Docker Application Container Engine  
  Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)  
  Active: active (running) since Tue 2016-11-29 14:12:54 UTC; 2h 47min ago  
    Docs: https://docs.docker.com  
  Main PID: 1204 (dockerd)  
     Tasks: 22  
    Memory: 82.9M  
      CPU: 1min 31.242s  
   CGroup: /system.slice/docker.service  
          └─1204 /usr/bin/dockerd -H fd://  
              ├─1212 docker-containerd -l unix:///var/run/docker/libcontainerd/docker-containerd.sock --shim docker-containerd-shim --metrics-interval=0 --start-ti  
Nov 29 16:29:33 ubuntu-xenial dockerd[1204]: time="2016-11-29T16:29:33.147992827Z" level=info msg="Layer sha256:6fc2e4ad81348c14fd2d7e3880b5db2bbc2f699a5cdcf18b  
Nov 29 16:29:36 ubuntu-xenial dockerd[1204]: time="2016-11-29T16:29:36.424426912Z" level=info msg="Layer sha256:a21398f45083710727f6f382cf232d5d50d4dc9589d6ff5  
Nov 29 16:30:09 ubuntu-xenial dockerd[1204]: time="2016-11-29T16:30:09.788820380Z" level=info msg="Layer sha256:9359188bd45e8c8b98fc0f0297ba87b4f5ed64e9dc8fbba7  
Nov 29 16:30:09 ubuntu-xenial dockerd[1204]: time="2016-11-29T16:30:09.913582571Z" level=info msg="Layer sha256:9359188bd45e8c8b98fc0f0297ba87b4f5ed64e9dc8fbba7  
Nov 29 16:30:28 ubuntu-xenial dockerd[1204]: time="2016-11-29T16:30:28.557526233Z" level=info msg="Layer sha256:be3b076c597ddebb0e264d732927a168c2e91c78516c84a1  
Nov 29 16:30:50 ubuntu-xenial dockerd[1204]: time="2016-11-29T16:30:50.018714100Z" level=info msg="Layer sha256:47e85df6c4115d2974ab00ef823c215917daea09011f4262  
Nov 29 16:30:56 ubuntu-xenial dockerd[1204]: time="2016-11-29T16:30:56.620202459Z" level=info msg="Layer sha256:ce06475a45c5830b905647755ce0324b8c313502bf66ea0c  
Nov 29 16:31:14 ubuntu-xenial dockerd[1204]: time="2016-11-29T16:31:14.829527550Z" level=info msg="Layer sha256:2b9628b0eba7b5a870fd354986be4d15e605d51c8f9f7f2a  
Nov 29 16:31:15 ubuntu-xenial dockerd[1204]: time="2016-11-29T16:31:15.102270791Z" level=info msg="Layer sha256:c8f15147c2663dc3d68f177fffc0934bb2c04e25eddcc6007  
Nov 29 16:32:33 ubuntu-xenial dockerd[1204]: time="2016-11-29T16:32:33.801824138Z" level=info msg="Layer sha256:72deb7ae364e53981d0d1befdf7d9c5b15208f93cd8e169b  
lines 1-22/22 (END)
```



Docker Union File System

AuFS ([AnotherUnionFS](#)) is a multi-layered filesystem that implements union mount

- Allows several filesystems or directories to be simultaneously mounted and visible through a single mount point
- Appears to be one filesystem to the end user



Example Docker Layers



Docker Layer – Each Docker image is composed of a series of layers. Docker uses Union File Systems to combine these into a single image. The combination of these layers gives the illusion of a traditional file system. Only the top layer is writeable.



Docker Hub

Docker's first-party cloud-based registry

- Hosts a broad repository of Docker images
- Contains nearly any popular open source technology
- Contains 'Official images' from vendors such as Canonical, Oracle, Red Hat, etc
- Provides **one free private Docker repo**, more for a paid subscription

hub.docker.com



Docker Private Registry

- The Registry is a stateless, highly scalable server side application that stores and lets you distribute Docker images.
- Private registries are registries that are hosted and managed by third party or the organization itself (not by Docker).



Running a Private Docker Registry

- docker run -d -p5000:5000 registry
- docker push localhost:5000/<id>

```
[ubuntu@instance-41:~/app$ sudo docker pull registry
Using default tag: latest
latest: Pulling from library/registry
3690ec4760f9: Already exists
930045f1e8fb: Pull complete
feaa90cbdbc: Pull complete
61f85310d350: Pull complete
b6082c239858: Pull complete
Digest: sha256:1152291c7f93a4ea2ddc95e46d142c31e743b6dd70e194af9e6ebe530f782c17
Status: Downloaded newer image for registry:latest
[ubuntu@instance-41:~/app$ sudo docker run -d -p5000:5000 registry
64730d7011f71d463d480982a765624b4f353e2f2c7003779281cf453c8b2178
```

```
[ubuntu@instance-41:~/app$ sudo docker push localhost:5000/plamar
The push refers to a repository [localhost:5000/plamar]
88c7ceeb5d69: Pushed
fa69d2e74ebd: Pushed
4b80ed184035: Pushed
90b244cdf6f8: Pushed
e8c1de3c7027: Pushed
011b303988d2: Pushed
latest: digest: sha256:4dd3b61159aeeda583f480c542da468a68c63633d7e8a51e5645609701814e72 size: 1572
```

<https://github.com/docker/docker.github.io/blob/master/registry/deploying.md>



Docker Registry Search

- Search Registry

➤ <https://docs.docker.com/registry/spec/api/#listing-repositories>

```
ubuntu@instance-41:~/app$ curl -X GET http://localhost:5000/v2/_catalog
{"repositories":["plamar"]}
```

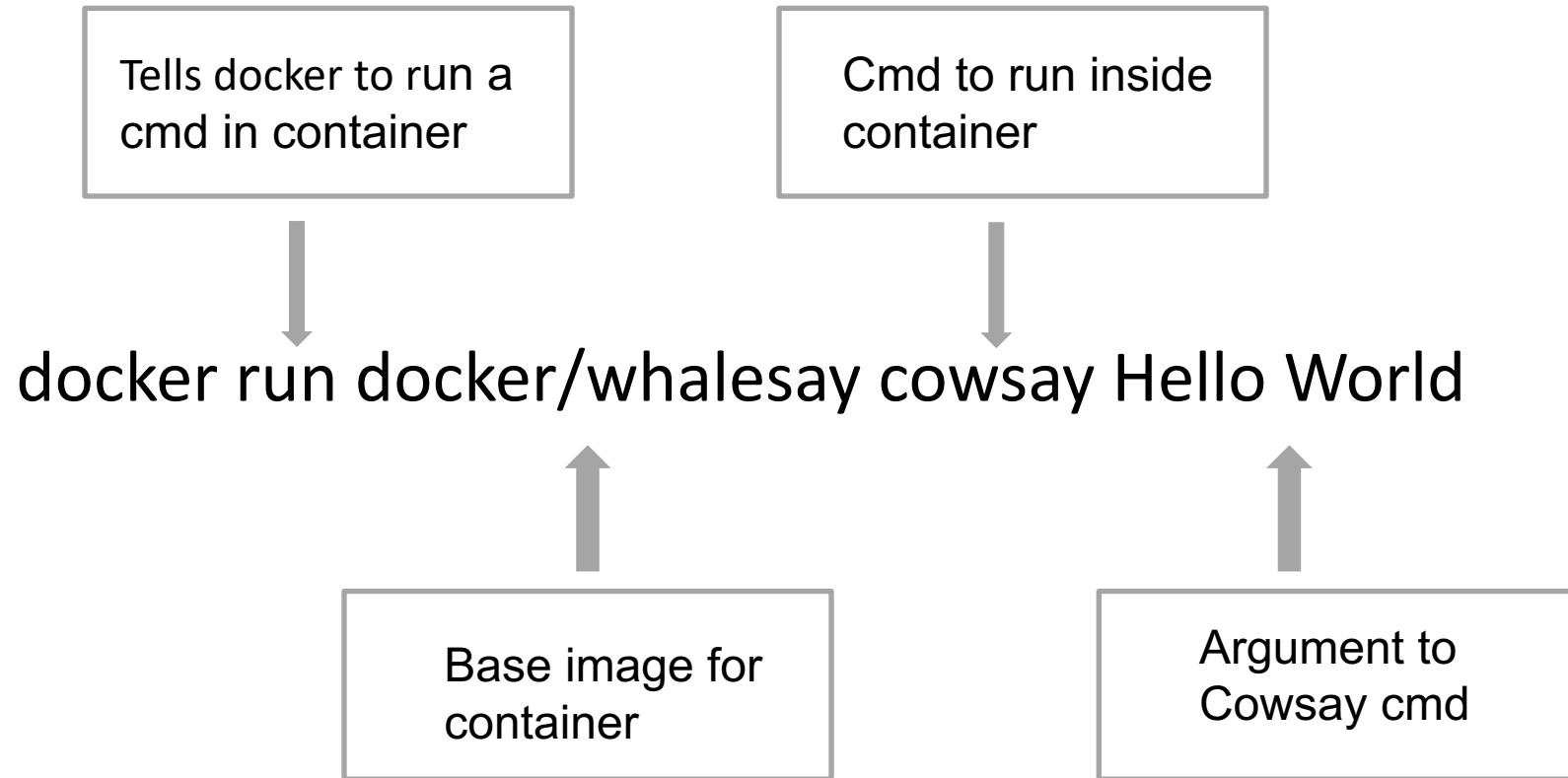
- List Image Tags

➤ <https://docs.docker.com/registry/spec/api/#listing-image-tags>

```
[ubuntu@instance-41:~/app$ curl -X GET http://localhost:5000/v2/plamar/tags/list
{"name":"plamar","tags":["latest"]}
```



Docker – Run Containers

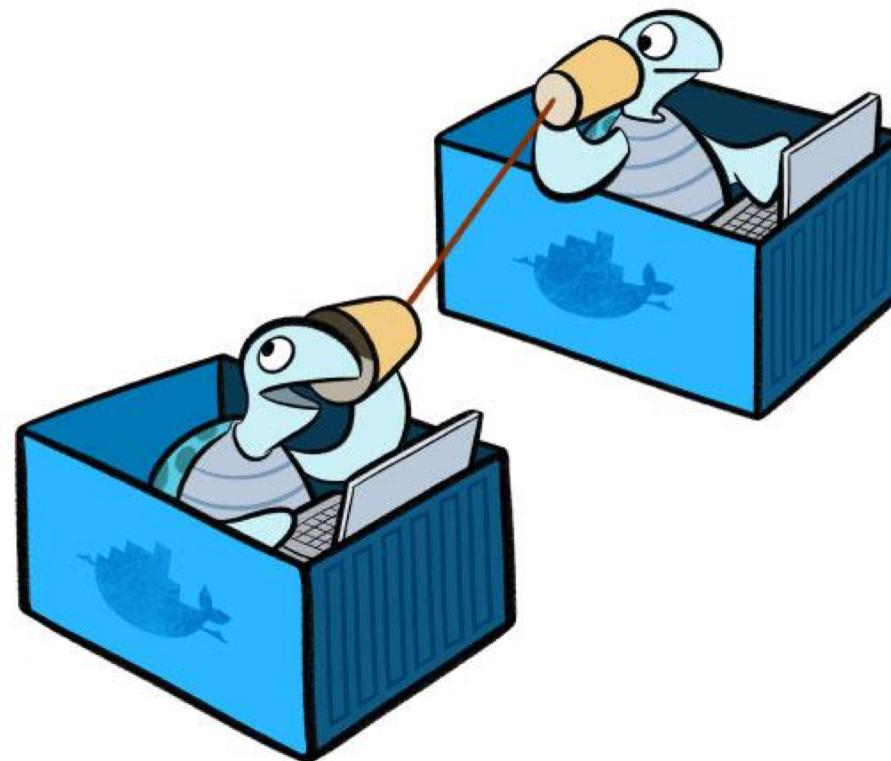


Docker – Other useful commands

- docker run -i -t <image> # interactive mode
- docker attach <container> # attach a process id to a container
- docker exec -it <container> /bin/bash # execute a cmd in container from host
- docker ps # list running containers
- docker ps -a # list all containers
- docker images # list all image
- docker stats # provide cpu, memory, etc. for containers
- docker pull # pull image from a repo
- docker inspect <container>|<image> # inspect an image or a container
- docker history <image> # check layers of an image
- docker container|image prune -f # remove unused containers or images



Docker Networking



Docker Networking Defaults

- Three networks are created by default

(bridge, host & none)

➤ **docker network ls**

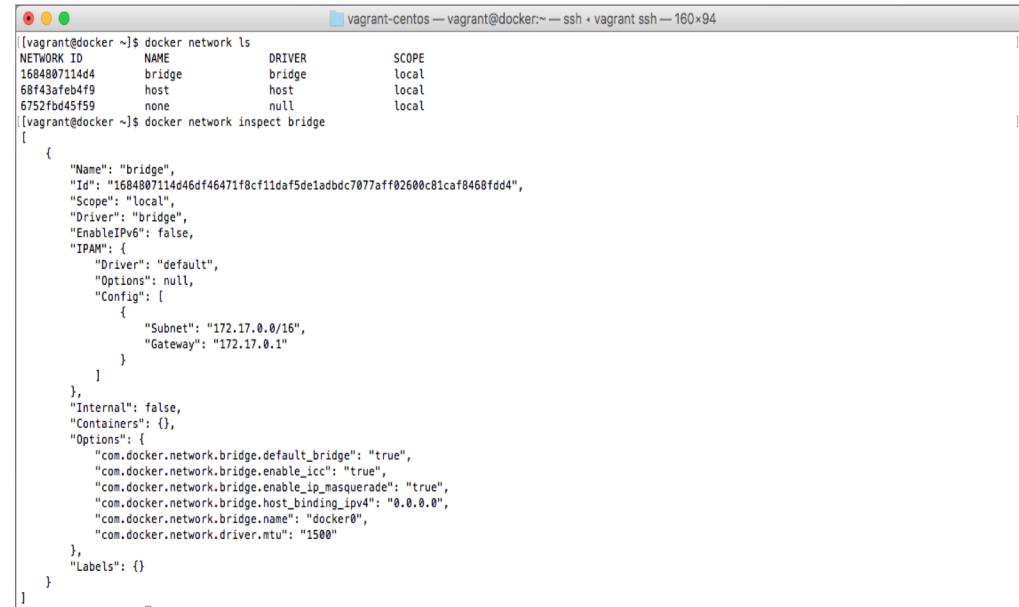
- We can easily inspect the bridge network

➤ **docker network inspect bridge**

- We see that a 172.17.0.0/16 address space

is allocated. Created Containers are given

an IP in this space by default.



```
[vagrant@docker ~]$ docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
1684807114d4   bridge    bridge      local
68f43afeb4f9   host      host       local
6752fb45f59   none     null       local

[vagrant@docker ~]$ docker network inspect bridge
[
  {
    "Name": "bridge",
    "Id": "1684807114d46df46471f8cf11daf5de1adbd7077aff02600c81caf8468fd4",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": null,
      "Config": [
        {
          "Subnet": "172.17.0.0/16",
          "Gateway": "172.17.0.1"
        }
      ]
    },
    "Internal": false,
    "Containers": {},
    "Options": {
      "com.docker.network.bridge.default_bridge": "true",
      "com.docker.network.bridge.enable_icc": "true",
      "com.docker.network.bridge.enable_ip_masquerade": "true",
      "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
      "com.docker.network.bridge.name": "docker0",
      "com.docker.network.driver.mtu": "1500"
    },
    "Labels": {}
  }
]
```



Docker Networking Defaults

```
vagrant@vagrant-centos:~$ docker run --name some-ghost -p 8080:2368 -d ghost
53e5d4de5054ca0b3cfaab91476f6901f373df3e4b215c2af855cc14be62a464
[vagrant@vagrant ~]$ docker network inspect bridge
[{"Name": "bridge",
 "Id": "1684807114d46df46471f8cf11daf5de1adbdc7077aff02600c81caf8468fdd4",
 "Scope": "local",
 "Driver": "bridge",
 "EnableIPv6": false,
 "IPAM": {
     "Driver": "default",
     "Options": null,
     "Config": [
         {
             "Subnet": "172.17.0.0/16",
             "Gateway": "172.17.0.1"
         }
     ]
 },
 "Internal": false,
 "Containers": {
     "53e5d4de5054ca0b3cfaab91476f6901f373df3e4b215c2af855cc14be62a464": {
         "Name": "some-ghost",
         "EndpointID": "e02d6b898b3d89ee192bbf228d467d49d8f811948a7abc79543524112148fa7f",
         "MacAddress": "02:42:ac:11:00:02",
         "IPv4Address": "172.17.0.2/16",
         "IPv6Address": ""
     }
 },
 "Options": {
     "com.docker.network.bridge.default_bridge": "true",
     "com.docker.network.bridge.enable_icc": "true",
     "com.docker.network.bridge.enable_ip_masquerade": "true",
     "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
     "com.docker.network.bridge.name": "docker0",
     "com.docker.network.mtu": "1500"
 },
 "Labels": {}
}
[vagrant@vagrant ~]$ ping 172.17.0.2
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.049 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.080 ms
64 bytes from 172.17.0.2: icmp_seq=3 ttl=64 time=0.183 ms
^C
--- 172.17.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.049/0.104/0.183/0.057 ms
```

- Created container is attached to bridge network by default
 - Can be attached to user created networks as well with **--network** flag
- Container is assigned 172.17.0.2 ip in the bridge network and appears in the docker inspect command
- The container can then be pinged at 172.17.0.2



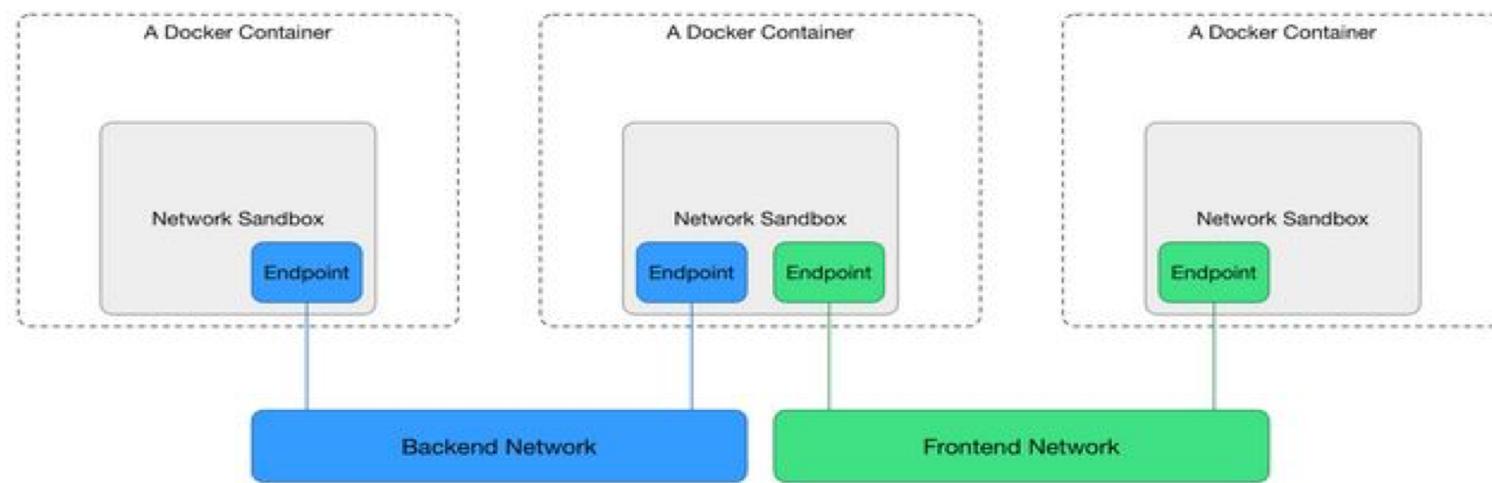
Docker Networking Commands

Command	Description
docker network create	Create a network
docker network ls	List networks
docker network inspect	Detailed information on network
docker network rm	Remove network
docker network disconnect	Disconnect container from network
docker network connect	Connect container to network



Isolated User Networks

- The bridge network is useful in a development environment
- Production environments usually require more security and isolated or more customized networking solutions.



Docker Networking – Lab

https://github.com/shekhar2010us/kubernetes_teach_git/blob/master/docker_networking/docker_networking_lab.md



docker0 bridge

- Docker0 bridge is the bridge network named *bridge* created automatically when you install Docker.
- Options.{com.docker.network.bridge.name}
- By default, the Docker server creates and configures the host system's docker0 a network interface called docker0, which is an ethernet bridge device. If you don't specify a different network when starting a container, the container is connected to the bridge and all traffic coming from and going to the container flows over the bridge to the Docker daemon, which handles routing on behalf of the container.



Customize docker0 bridge

- You can configure the default bridge network's settings using flags to the **dockerd** command. However, the recommended way to configure the Docker daemon is to use the `daemon.json` file, which is located in `/etc/docker/` on Linux
- Restart Docker after making changes to the `daemon.json` file

```
{  
  "bip": "192.168.1.5/24",  
  "fixed-cidr": "192.168.1.5/25",  
  "fixed-cidr-v6": "2001:db8::/64",  
  "mtu": 1500,  
  "default-gateway": "10.20.1.1",  
  "default-gateway-v6": "2001:db8:abcd::89",  
  "dns": ["10.20.1.2", "10.20.1.3"]  
}
```

- `--bip=CIDR`: supply a specific IP address and netmask for the docker0 bridge, using standard CIDR notation.
- `--fixed-cidr=CIDR` and `--fixed-cidr-v6=CIDRv6`: restrict the IP range from the docker0 subnet, using standard CIDR notation.
- `--mtu=BYTES`: override the maximum packet length on docker0.
- `--default-gateway=Container default Gateway IPV4 address` and `--default-gateway-v6=Container default gateway IPV6 address`: designates the default gateway for containers connected to the docker0 bridge, which controls where they route traffic by default.
- `--dns=[]`: The DNS servers to use



Docker Volumes



Agile Brains Consulting

©Agile Brains Consulting Inc. All rights reserved. Not to be reproduced without prior written consent.



Volume Persistence

{docker -v} flag can mount the volume to a specific directory

Volumes can be mounted using absolute path or specified “Docker Volumes” which resides in `/var/lib/docker/...`



Docker Volume Commands

Command	Description
<code>docker volume create</code>	Create a volume
<code>docker volume ls</code>	List volumes
<code>docker volume inspect</code>	Detailed information on volume
<code>docker volume rm</code>	Remove volume



Working with Volumes

```
[ubuntu@instance-41:~$ docker volume create --name myvolume  
myvolume  
[ubuntu@instance-41:~$ docker volume ls  
DRIVER          VOLUME NAME  
local           030f652ad7ea6766f3d17a421233896b78296cf89d9bc816c69e6084df1aa24d  
local           94ed6b916617b932638619fcbdbfc77d843175a57c520d02d3162f5750916409  
local           myvolume  
[ubuntu@instance-41:~$ docker volume inspect myvolume  
[  
  {  
    "Name": "myvolume",  
    "Driver": "local",  
    "Mountpoint": "/var/lib/docker/volumes/myvolume/_data",  
    "Labels": {},  
    "Scope": "local"  
  }  
]  
[ubuntu@instance-41:~$ docker volume rm myvolume  
myvolume -
```



Dangling Volumes

- If container is deleted with a volume attached, the volume remains. Sometimes removing all such ‘dangling’ volumes is desired

```
$ docker volume prune
```



Docker Daemon Logging



Docker Daemon Logging

- Docker includes multiple logging mechanisms to help you get information from running containers and services. These mechanisms are called logging drivers.
- Each Docker daemon has a default logging driver, which each container uses unless you configure it to use a different logging driver.
- To configure the Docker daemon to default to a specific logging driver, set the value of log-driver to the name of the logging driver in the *daemon.json* file, which is located in */etc/docker/* on Linux hosts



Docker Daemon Logging

- The daemon logs may help you diagnose problems. The logs may be saved in one of a few locations:

Operating System	Location
RHEL, Oracle Linux	/var/log/messages
Debian	/var/log/daemon.log
Ubuntu 16.04+, CentOS	Use the command journalctl -u docker.service
Ubuntu 14.10-	/var/log/upstart/docker.log
macOS (Docker 18.01+)	~/Library/Containers/com.docker.docker/Data/vms/0/console-ring
macOS (Docker <18.01)	~/Library/Containers/com.docker.docker/Data/com.docker.driver.amd64-linux/console-ring
Windows	AppData\Local



Docker Daemon Logging

- To enable debugging, the recommended approach is to set the **debug** key to true in the *daemon.json* file, which is usually located in /etc/docker/
- On macOS or Windows, do not edit the file directly. Instead, go to **Preferences / Daemon / Advanced**.

Change log driver to syslog →

```
{ "log-driver": "json-file", or syslog  
  "log-opt": {  
    "max-size": "10m",  
    "max-file": "3",  
    "labels": "production_status",  
    "env": "os,customer"  
  }  
}
```



Container Logging

- To find the current default logging driver for the Docker daemon:

```
docker info --format '{{.LoggingDriver}}
```

- When you start a container, you can configure it to use a different logging driver than the Docker daemon's default:
 - *docker run -it --log-driver <driver> <image> <params>*



Container Logging – Lab

https://github.com/shekhar2010us/kubernetes_teach_git/tree/master/docker_logging

