Distributed Systems Study Group

First Meeting: Dec 3rd, 2019

Gerry Seidman
gerry@iamx.com

Distributed Systems Study Group

- Get a deeper understanding of Distributed Systems
    - Architectural Patterns
    - Today’s “standard” technology
    - Technology ‘Internals’

- Collaborate with others to get Software installed/running

- Design and Build a Distributed Application from Scratch

- Maintain a public/shared Journal of what we have learned

Who are we?

- Developer
- Architect
- Admin
- SRE
- Application Security
- Data Scientist

This is a Study Group

- Everyone has something to contribute
    - Prepare and Give presentations
    - Help others in the group with their work
    - Take/Post notes to be posted onto our Shared Journal (git repo)
- Format
    - Hands-on / Networking
    - Questions for the Group
    - Presentation
    - Hands-on
- Not everyone will go the same speed
    - Each time we meet there will be hands-on time to help people catch up
    - You should always be able to follow the presentations

Areas that We can Cover

- Containers / Kubernetes
  - Application packaging, deployment and management
- Kafka
  - Used for building real-time data/event pipelines and streaming apps.
- Prometheus
  - Time Series Database used for event monitoring and alerting
- Grafana
  - Front end Dashboards for Monitoring and Analytics
- CI/CD (Jenkins, other)
  - Continuous Integration / Continuous Delivery
- Distributed Tracing
  - Used for building real-time data/event pipelines and streaming apps.
- Hadoop/Spark
- ELK Stack –Elastic Logstash Kibana
- Agile
  - JIRA

And for all of them…

Security … oh Security

Goals for Hands-On Tonight

- VMs (or external machines) set up with the appropriate network inter-connectivity
- Install Linux (Headless, ie no desktop interface, only ssh)
- Have docker properly installed
- Have a kubernetes cluster installed

It Takes a Village

- Sharing/memorializing our experience

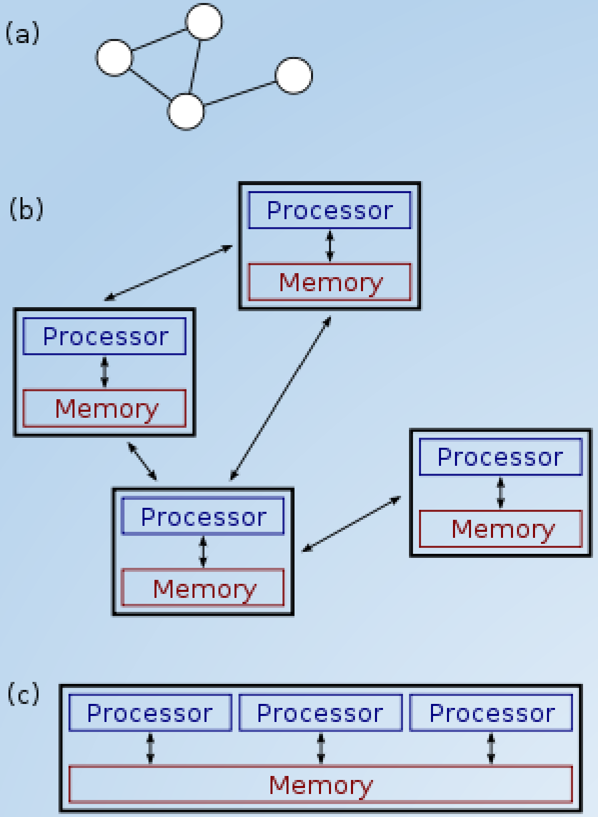
https://github.com/GerrySeidman/Distributed-Systems-Study-Group

Agenda (approximately)

- Overview of Distributed Systems
  - What we’d like to build
- Quick Overview
  - File Systems
  - Networking
- Understanding Containers
- Understanding Kubernetes

What is a Distributed System

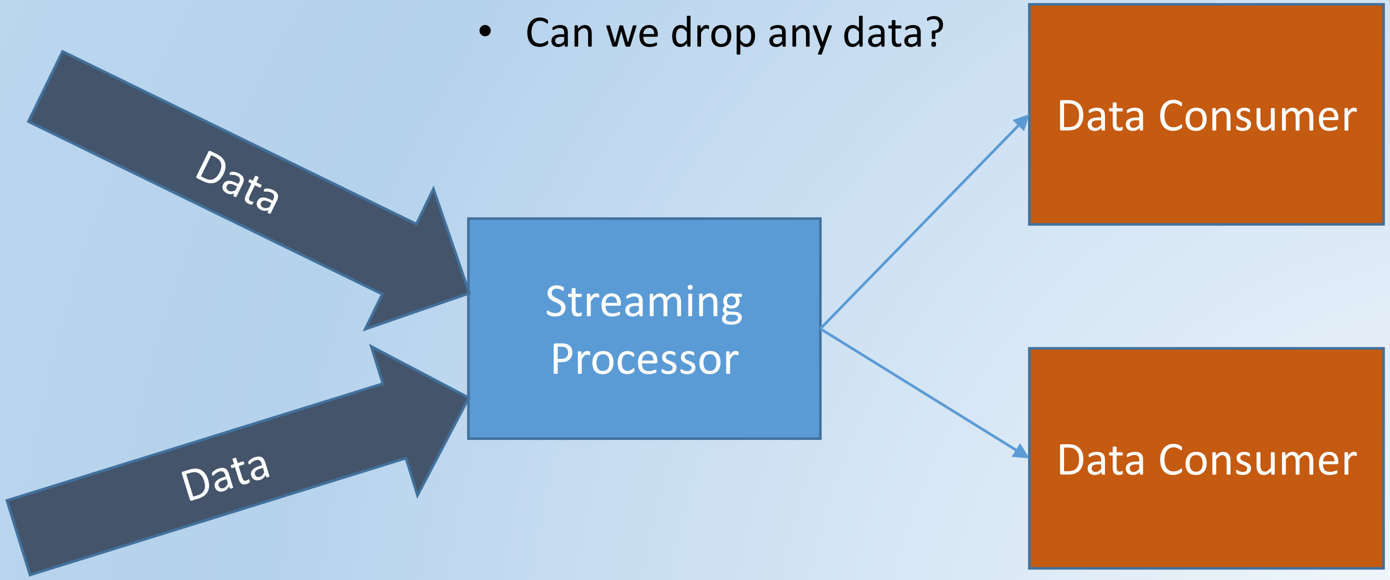
- Cluster Service
  - Database
  - Message Bus
  - HTTP Server
- Complex System
  - Many of the above
  - Working Together (loosely coupled)
  - Possibly spanning multiple data centers
    - Either Enterprise or Cloud
    - Buzz Words: Private Cloud, Multi-Cloud, Hybrid Cloud



For Example: How to handle Streaming data?

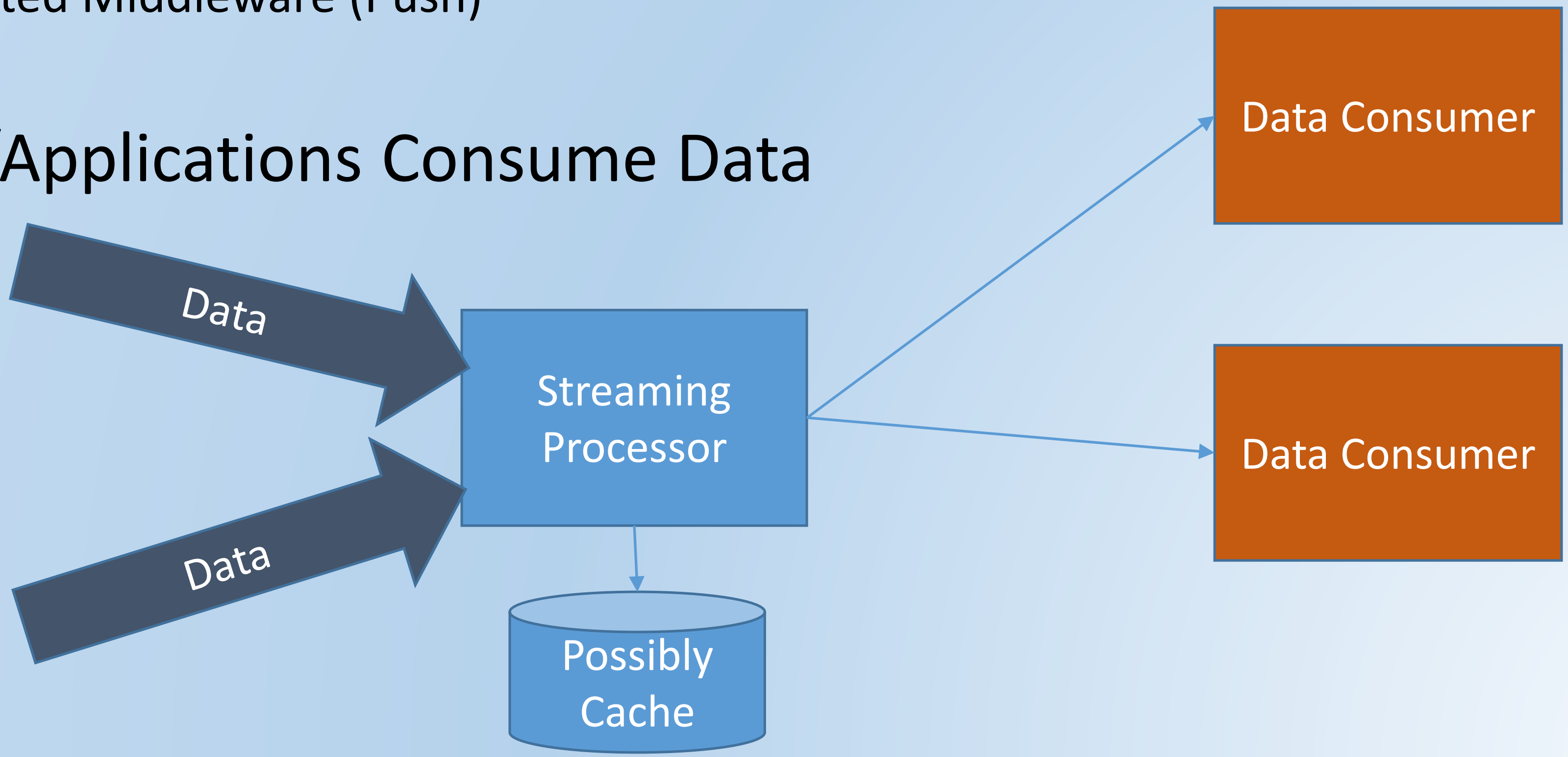
- Sources
  - Twitter Feeds
  - Stock Trades
  - Logs
  - SIEMS
- Possibly many
  - Homogeneous
  - Heterogenous
- Reliability
  - Availability
  - Scale on inbound data
    - Sustained or Bursty
  - Scale on # of Consumers
  - Can we drop any data?

- Consumers
  - Homogeneous
  - Heterogeneous
- Reliability
  - Scale on data that I want
  - Can we drop any data?

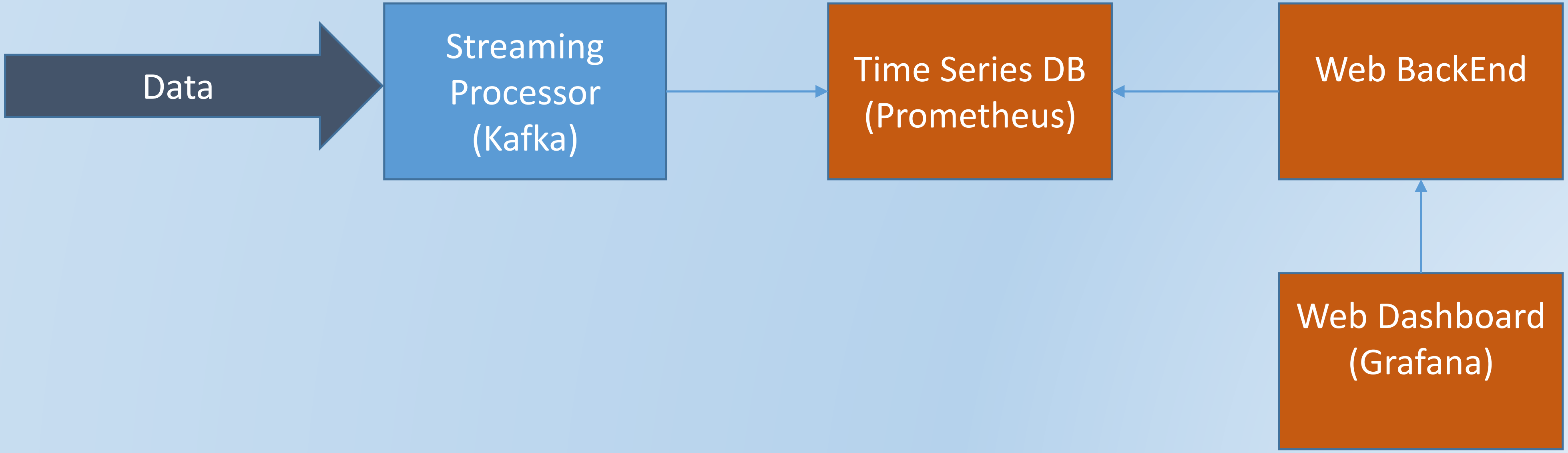


Streaming Data Processing

- Cluster of Streaming processors receive data
  - Persistence Model and Guarantees
  - Flavors
    - Message Oriented Middleware (Push)
    - Kafka (Pull)
- Other Processes/Applications Consume Data



Complex Distributed System Example


Data

Streaming Processor (Kafka)

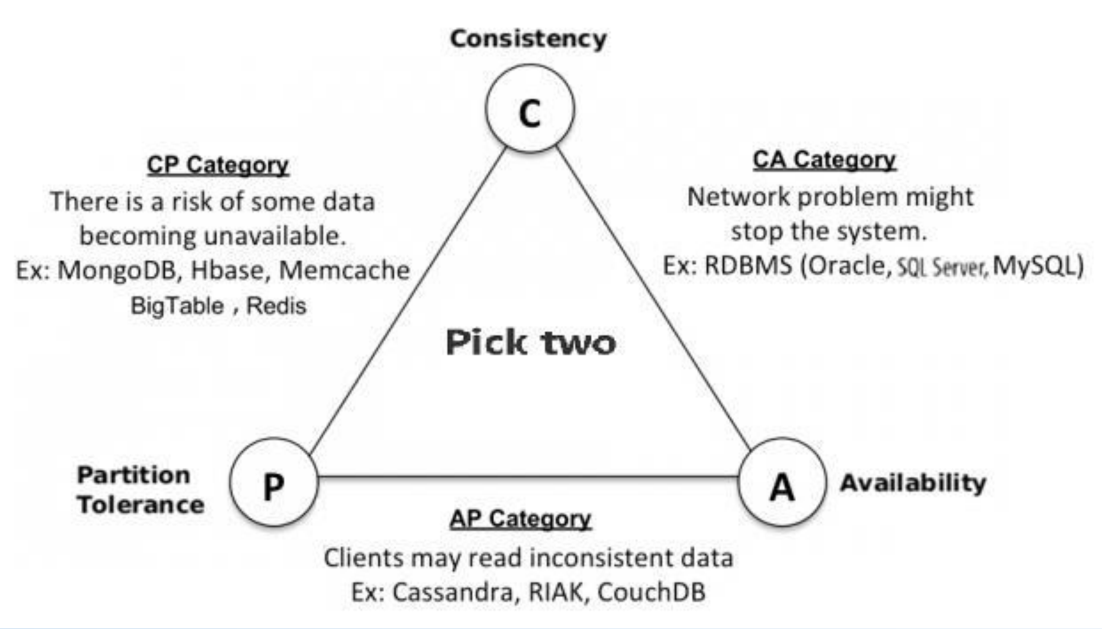
Time Series DB (Prometheus)

Web BackEnd

Web Dashboard (Grafana)

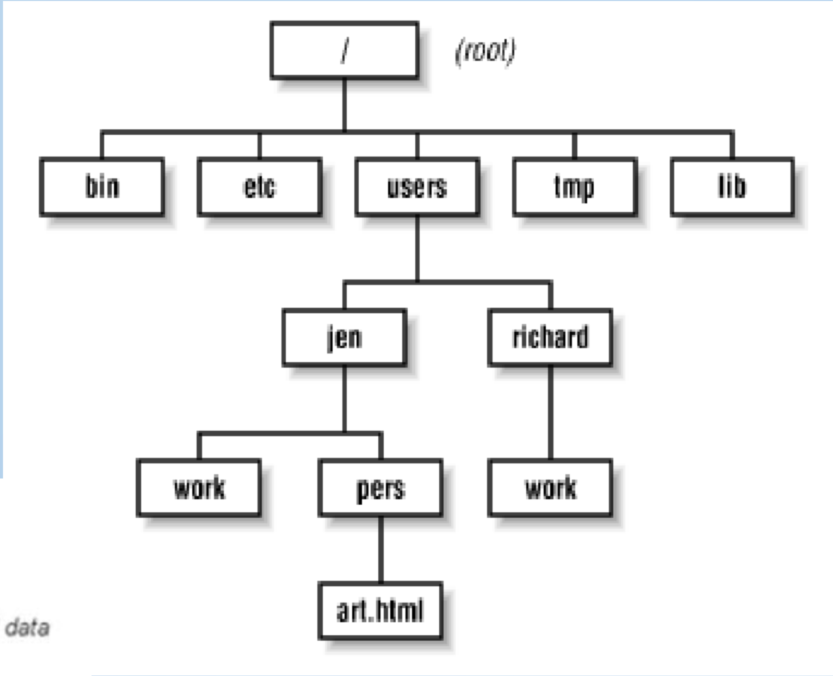
CAP Theorem - Tradeoffs

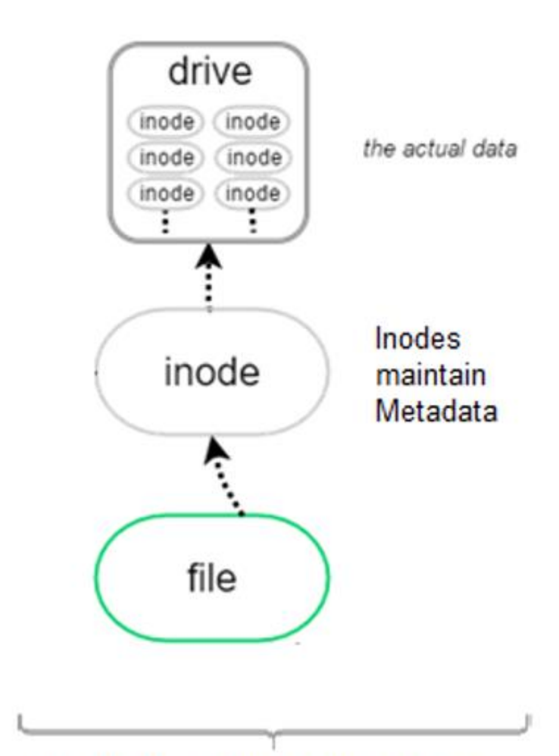
- Consistency:
    - Every read receives the most recent write or an error
- Availability:
    - Every request receives a (non-error) response, without the guarantee that it contains the most recent write
- Partition tolerance:
    - The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes


Consistency
C
CP Category
There is a risk of some data becoming unavailable.
Ex: MongoDB, Hbase, Memcache BigTable , Redis
CA Category
Network problem might stop the system.
Ex: RDBMS (Oracle, SQL Server, MySQL)
Pick two
Partition Tolerance
P
A
Availability
AP Category
Clients may read inconsistent data
Ex: Cassandra, RIAK, CouchDB

File Systems

- Volume (Logical/Physical Drives)
  - HD, SSD, CD, USB, Memory, Network
- Directory Structure
- File System
  - Linux EXT4, XFS, ZFS
  - Windows: FAT, NTFS
- Inodes
- Metadata + Extended Attributes

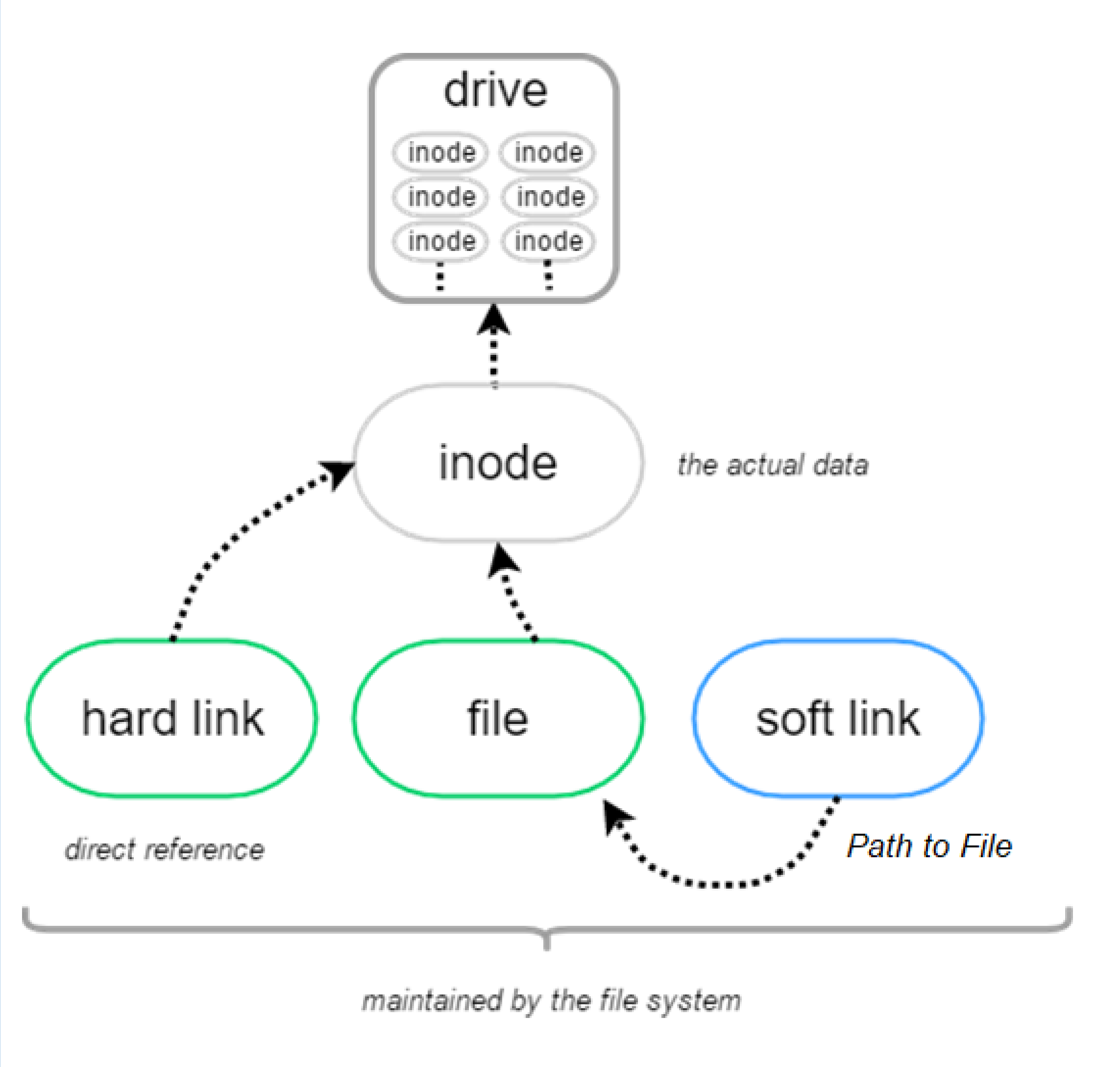




File System Maintains Directory and File Meta-Data

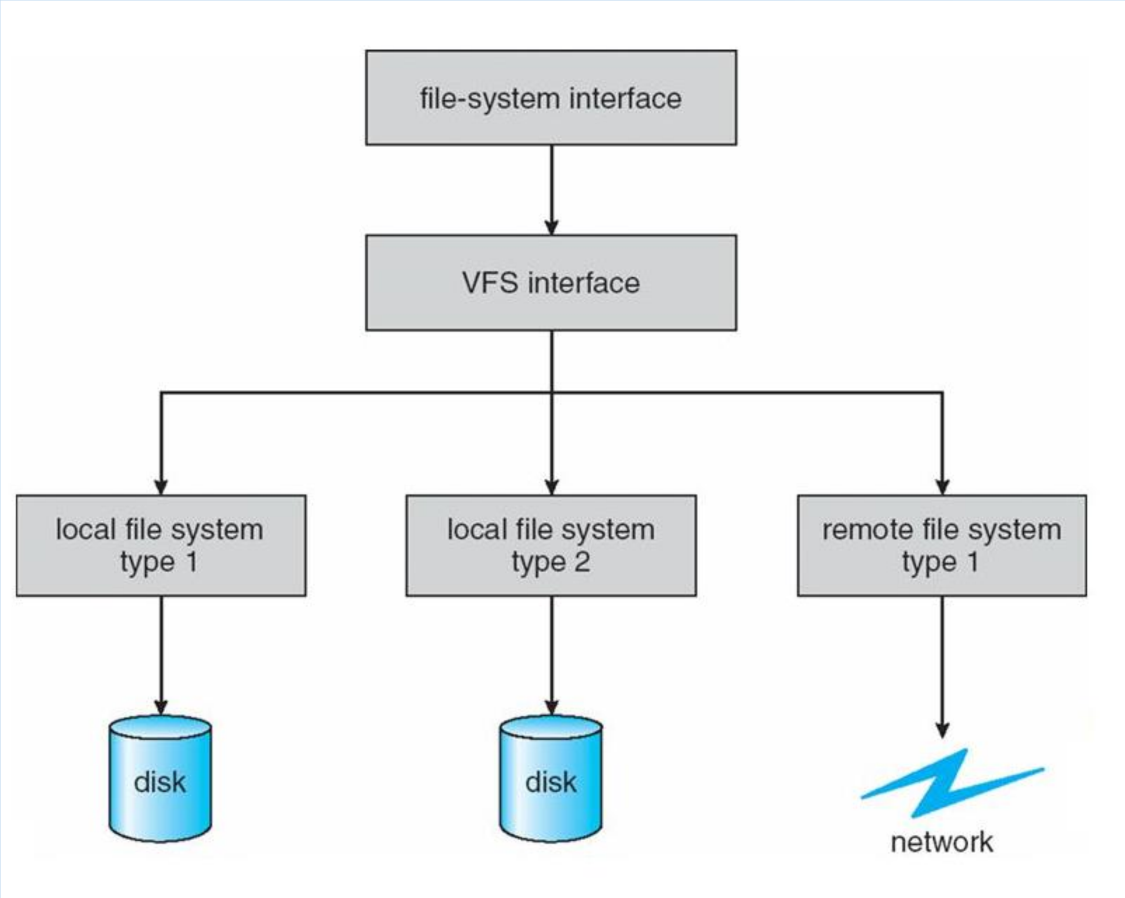
Links

- Hard
- Soft/Symbolic
- Works for Files or Directories



Virtual File System (VFS)

- Mounts
  - ‘Mount’ a FileSystem to a directory path in the VFS



Overlay File System

- We’ll get Back to this with Containers….

- ~~Copy on Write~~
- ~~https://docs.docker.com/storage/~~
- ~~https://docs.docker.com/storage/storagedriver/overlayfs-driver/~~

I/O Basics

- Stream I/O
  - File
  - Pipe
  - Sockets
- Random Access I/O
  - Typically things like databases
  - But Databases may use raw Volumes directly

IP Addresses and DNS

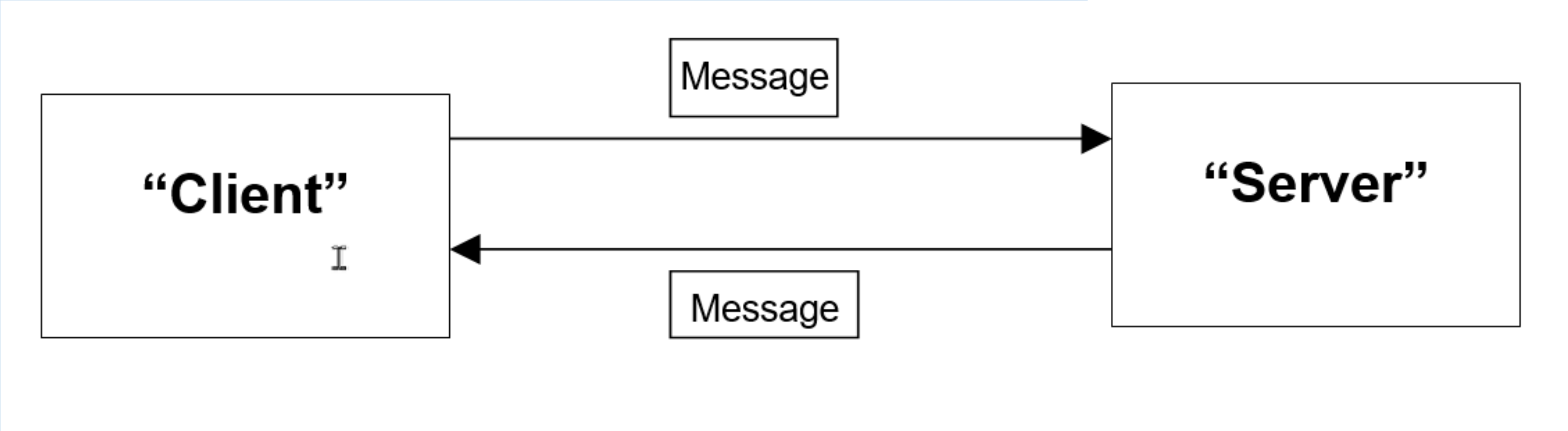
- IP Address
- DNS
- Subnet
  - Classless Inter-Domain Routing (CIDR)
    - Superseded Class only Routing ~1981 A/B/C/D
  - Subnet Designation Notation
    - 192.168.16.0/24
    - Alternatively
      - Network Mask: 255.255.255.0

Socket Basics

- An “server” application
  - Can register a ‘Service’
  - At a Port (1-65535)
  - On one of its Network Interface
  - Which can then ‘accept’ connections
  - Upon accept it has a Socket

- A “client” application
  - Can connect to a ‘Service’
  - On a Network Interface
  - At a Port (1-65535)
  - Upon Connect it has a Socket

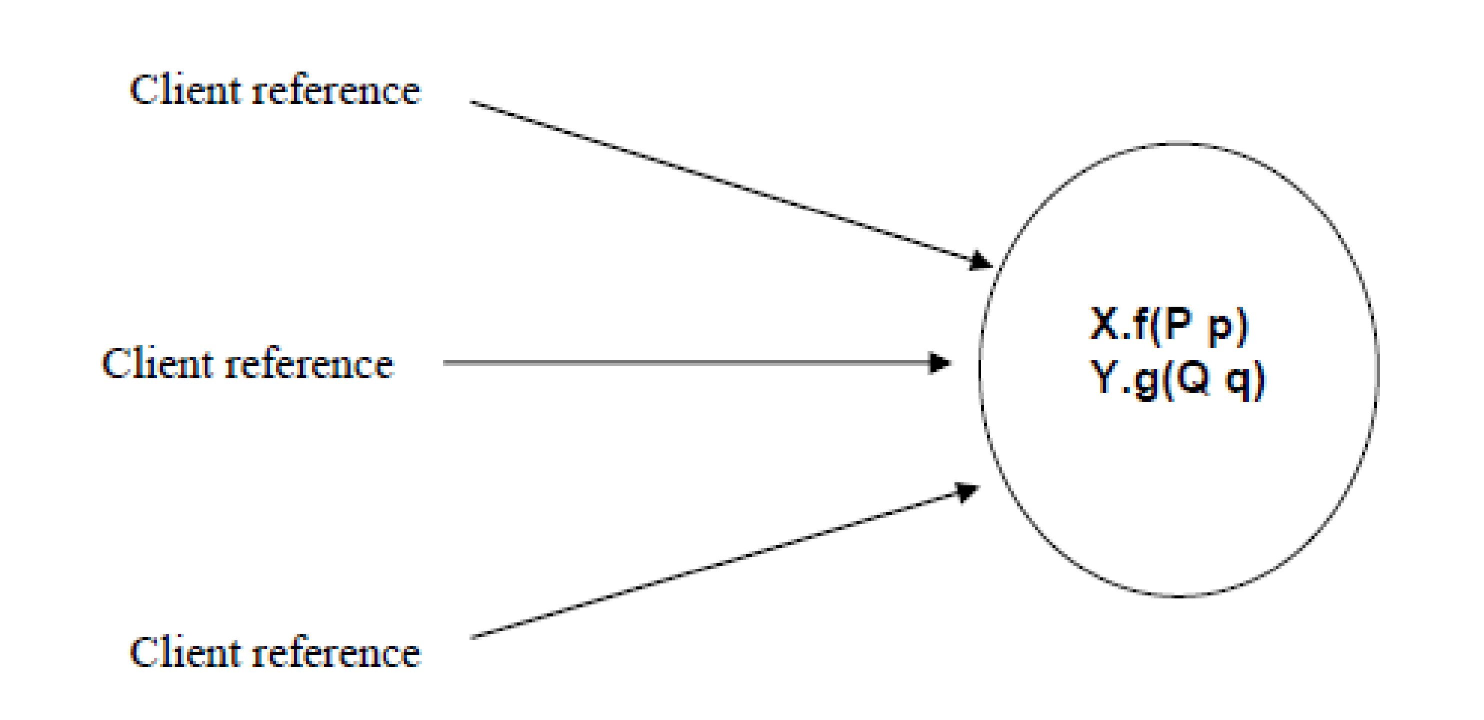
- Each Side has a Socket
  - From then on it is all about ‘Protocol



IP and TCP/IP

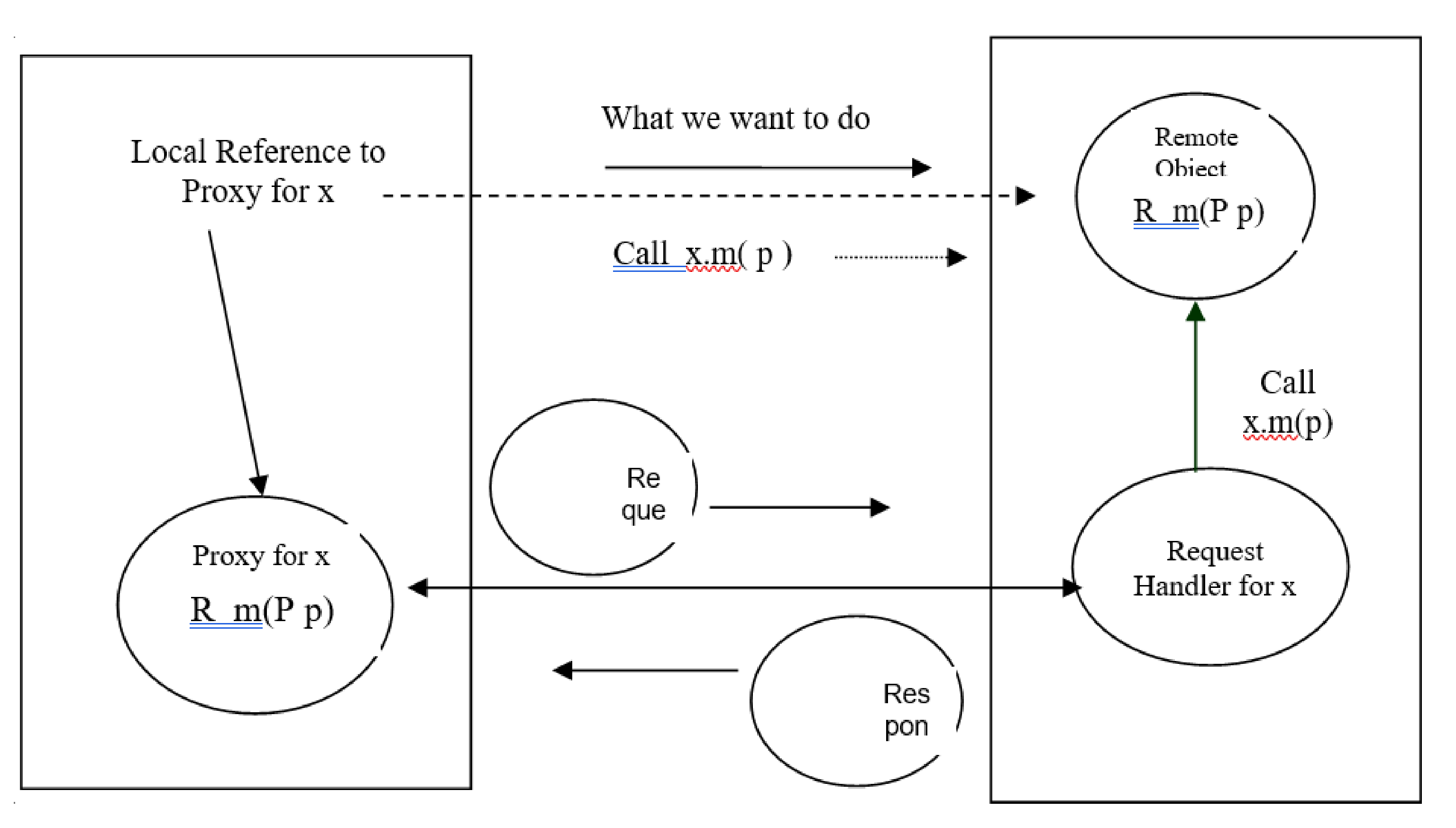
- IP Packets
  - Source Address/Port ➔Destination Address/Port + Payload
  - TCP/IP and UDP (others like ICMP for Ping)
- TCP Sockets
  - Frame Buffer
  - Writes can block

Calling a Function/Method on an Object

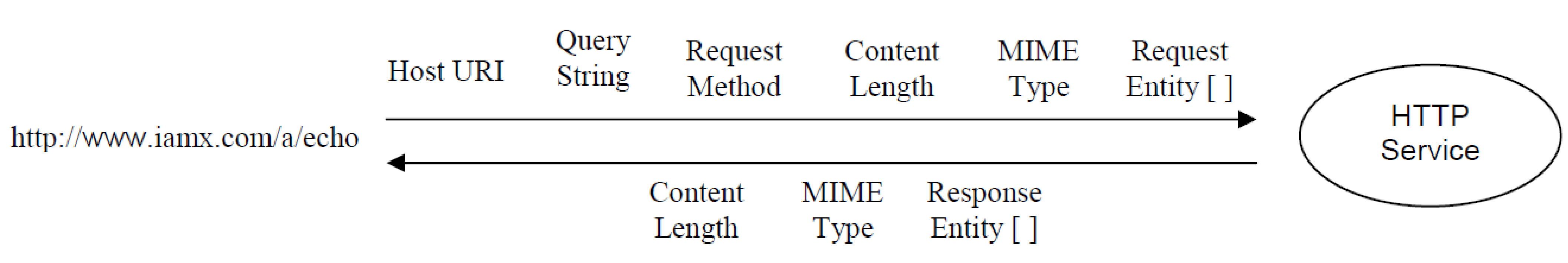


Remote Procedure Calls

- RPC Protocols
  - REST
  - gRPC
  - CORBA



- Example REST over HTTP



Understanding HTTP Sessions

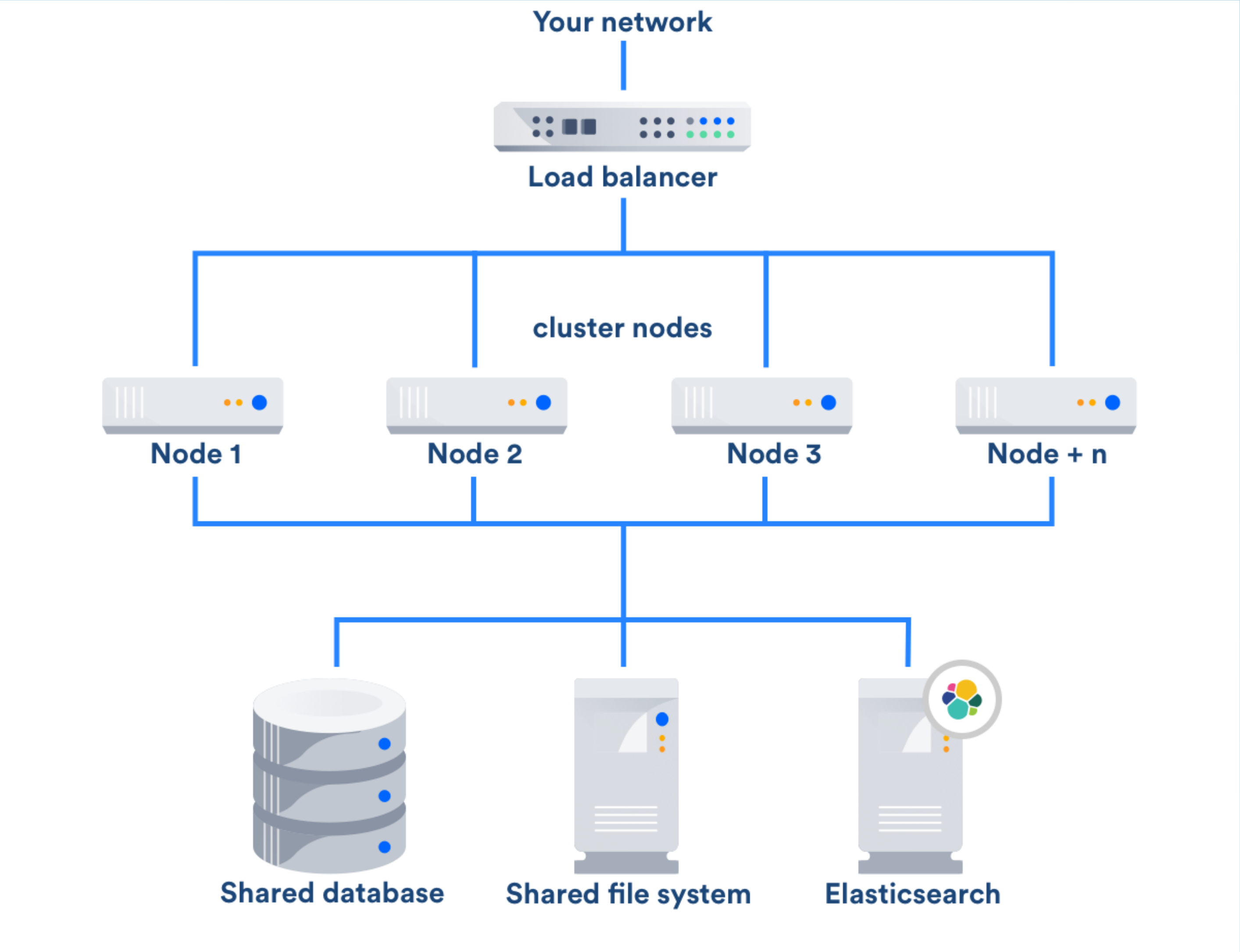
- curl -v --cookie "Elf=Keebler" "http://www.reunion.allaire.com/somepath?dog=Lassie&cow=Elsie"

```
* Connected to www.reunion.allaire.com (198.185.159.144) port 80 (#0)
> GET /somepath?dog=Lassie&cow=Elsie HTTP/1.1
> Host: www.reunion.allaire.com
> User-Agent: curl/7.55.1
> Accept: */*
> Cookie: Elf=Keebler
>
< HTTP/1.1 404 Not Found
< date: Tue, 03 Dec 2019 20:58:40 GMT
< expires: Thu, 01 Jan 1970 00:00:00 GMT
< content-type: text/html;charset=utf-8
< Age: 4
< Set-Cookie: crumb=BXMfdu9zkjHqNmIzNGFhYWQ1NGIzZmU5YWFiNGNhNmU0NjVmYmM0;Path=/
< Content-Length: 2052
< x-contextid: 9toQ3jNh/GuREc2eU
< server: Squarespace
```

NAT, Firewalls, IP Tables

- Network Address Translation (NAT)
  - Uses a single external IP Address
  - Internally provides a Subnet (ie many IP addresses from one)
- IP Tables
  - Redirect or block packets
- Firewall Rules

Load Balancers and Content Switches

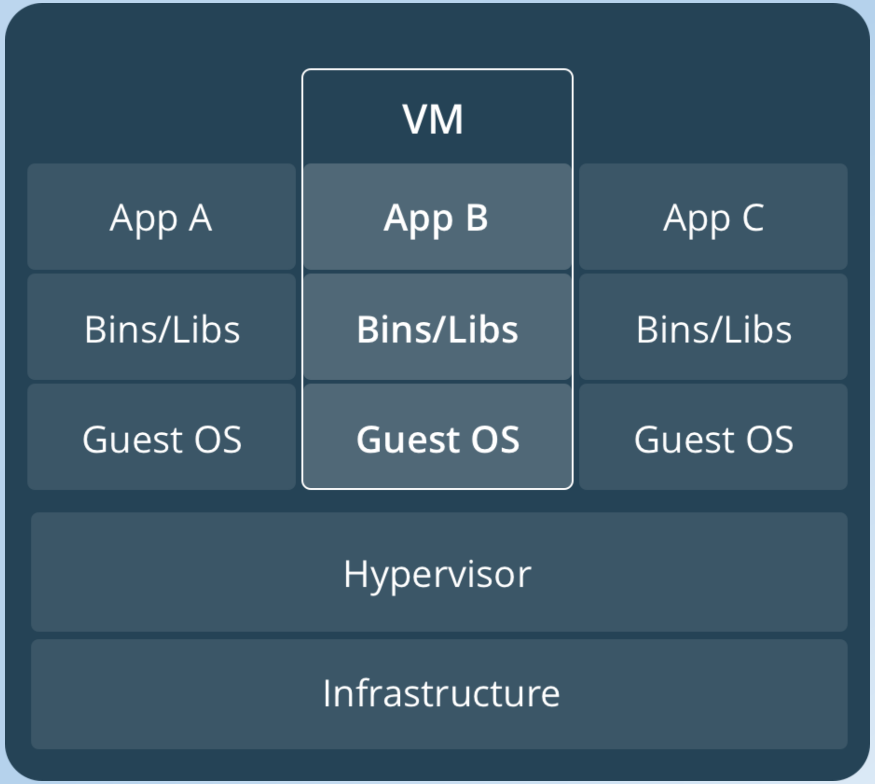


Software Packaging and Deployment

- Developer writes some code, tests it locally and we’re ready to deploy
  - But where?
  - How to deliver it?
- Zip it up and install it on some dedicated machine
  - Oh yeah.. There’s also configuration files
- You need a machine to handle peak load of that machine
  - How much Disk? RAM? Network? GPU? Fast Processor?
  - Hard Allocation of Resources

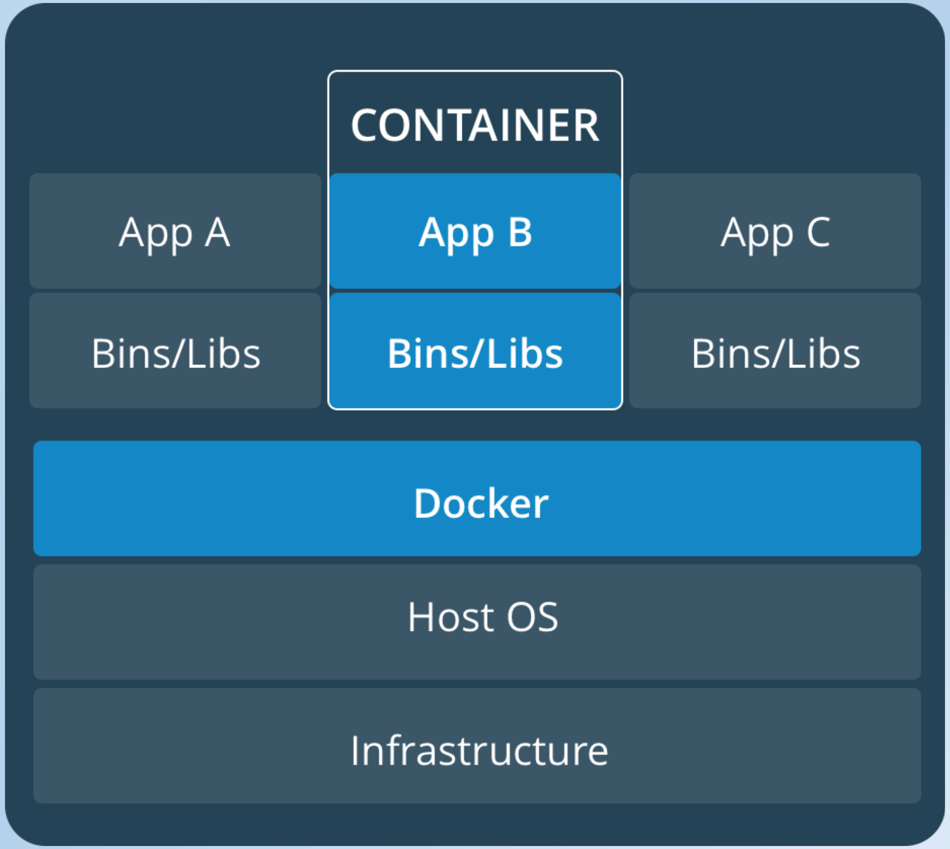
Virtual Machine

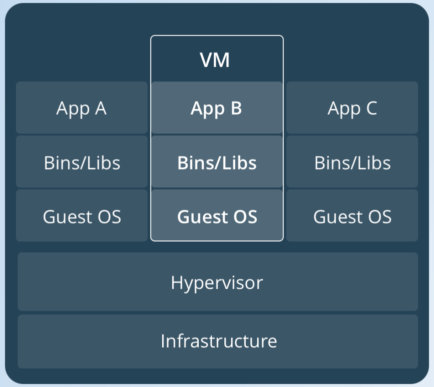
- Unit of deployment a VM Image
  - Just a big TAR file
  - When running it is a VM
- Semi-Soft Allocation of Resources
  - Disk, Memory, Network
- Slow to load
- Strong Security Boundary
- Hard to maintain
  - Same amount of work as a dedicated Machine
  - Security patches



Container

- Unit of deployment a Container Image
  - Just a big TAR file
  - When running it is a Container
- Soft Allocation of Resources
  - Disk, Memory
  - Some shared across Containers
- Fast to load
  - Run as a Host Process
- Pretty Strong Security Boundary
  - Linux Kernel
- Easier to maintain
  - Host OS





What is Docker

- Docker File Format for Container Images

- Docker Container Image builder
    - Must have Root access to build

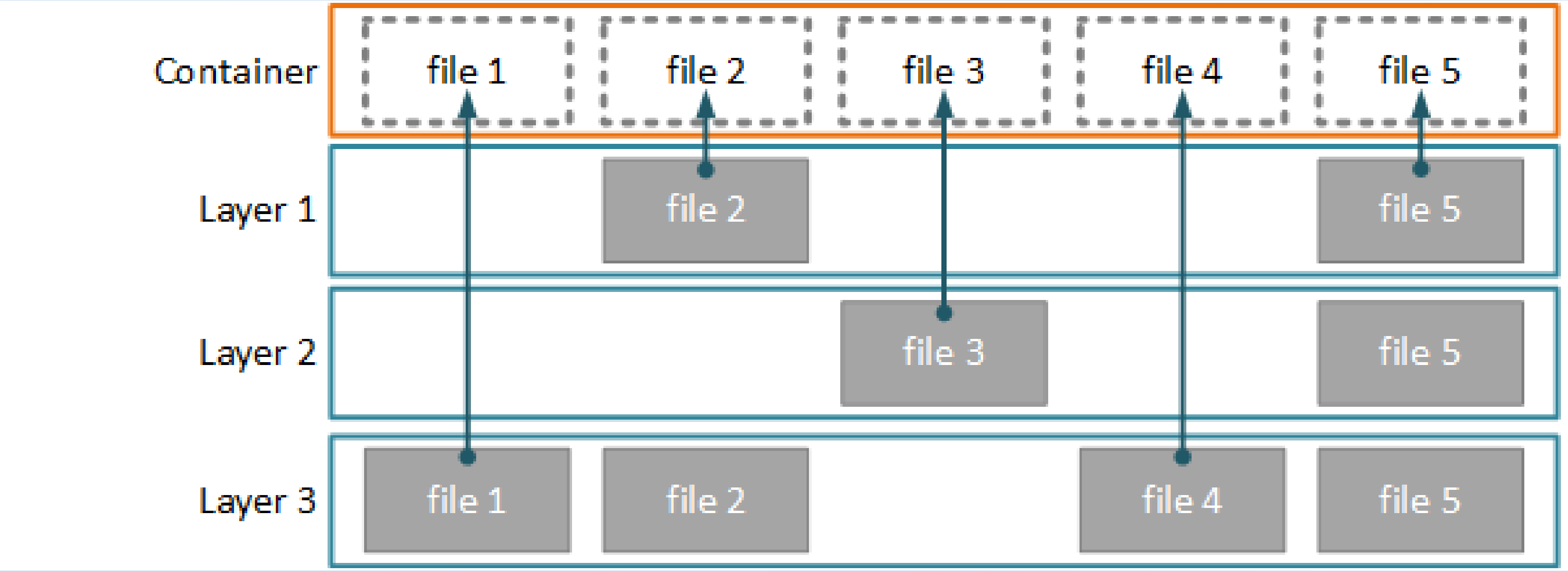
- Docker Runtime ‘server’
    - dockerd
        - REST or CLI Client
    - Must have Root access to launch containers

- Docker Registry

- Other Container implementations
    - Singularity
    - Podman/Buildah

Overlay File System – Container Image Layers

- Copy on Write File System



Container	file 1	file 2	file 3	file 4	file 5
Layer 1		file 2			file 5
Layer 2			file 3		file 5
Layer 3	file 1	file 2		file 4	file 5

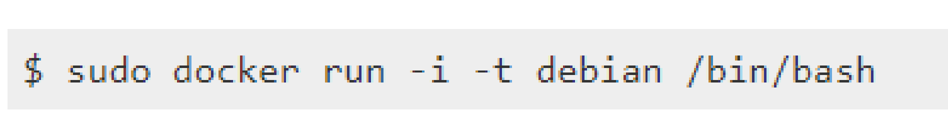
Container Resource Management

- Container Isolation
  - Control Groups (cgroups)
  - Namespaces

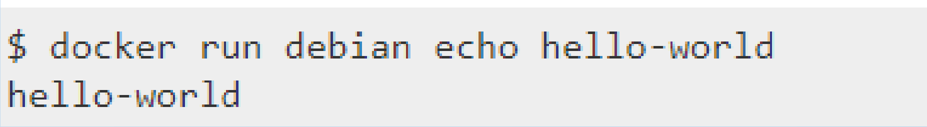
Namespaces

- The pid namespace: Process isolation (PID: Process ID).
- The net namespace: Managing network interfaces (NET: Networking).
- The ipc namespace: Managing access to IPC resources (IPC: InterProcess Communication).
- The mnt namespace: Managing filesystem mount points (MNT: Mount).
- The uts namespace: Isolating kernel and version identifiers. (UTS: Unix Timesharing System).

Docker Simplest Example



```
$ sudo docker run -i -t debian /bin/bash
```



```
$ docker run debian echo hello-world
hello-world
```

- http://containertutorials.com/

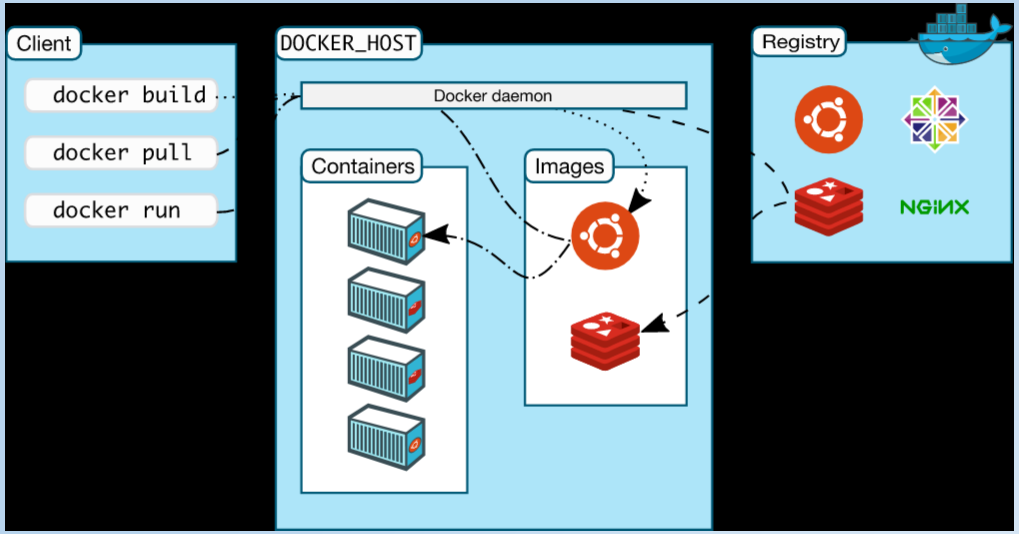
Docker Notes

- Docker Object
  - Images
    - docker build
  - Containers
    - docker run -i -t ubuntu /bin/bash

- Will pull from configured registries

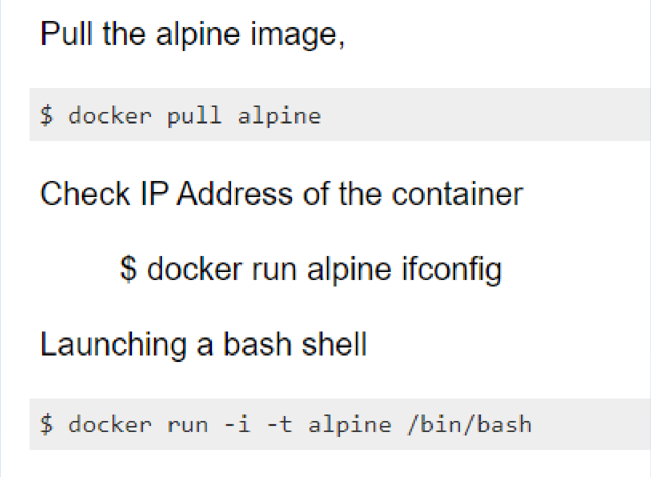
- Repositories and Registries
  - Git analogy (repo vs github)
    - The thing to remember here is a Docker repository is a place for you to publish and access your Docker images. Just like GitHub is a place for you to publish and access your git repos
  - Push/pull

- Need to do everything as root



Understanding Images

- Registry vs local Repository
  - Public (dockerhub) vs Private registries
  - docker search alpine-apache
- docker images
  - Image variants [image]:[tag]
  - docker image
- docker build
- docker pull



Pull the alpine image,

```
$ docker pull alpine
```

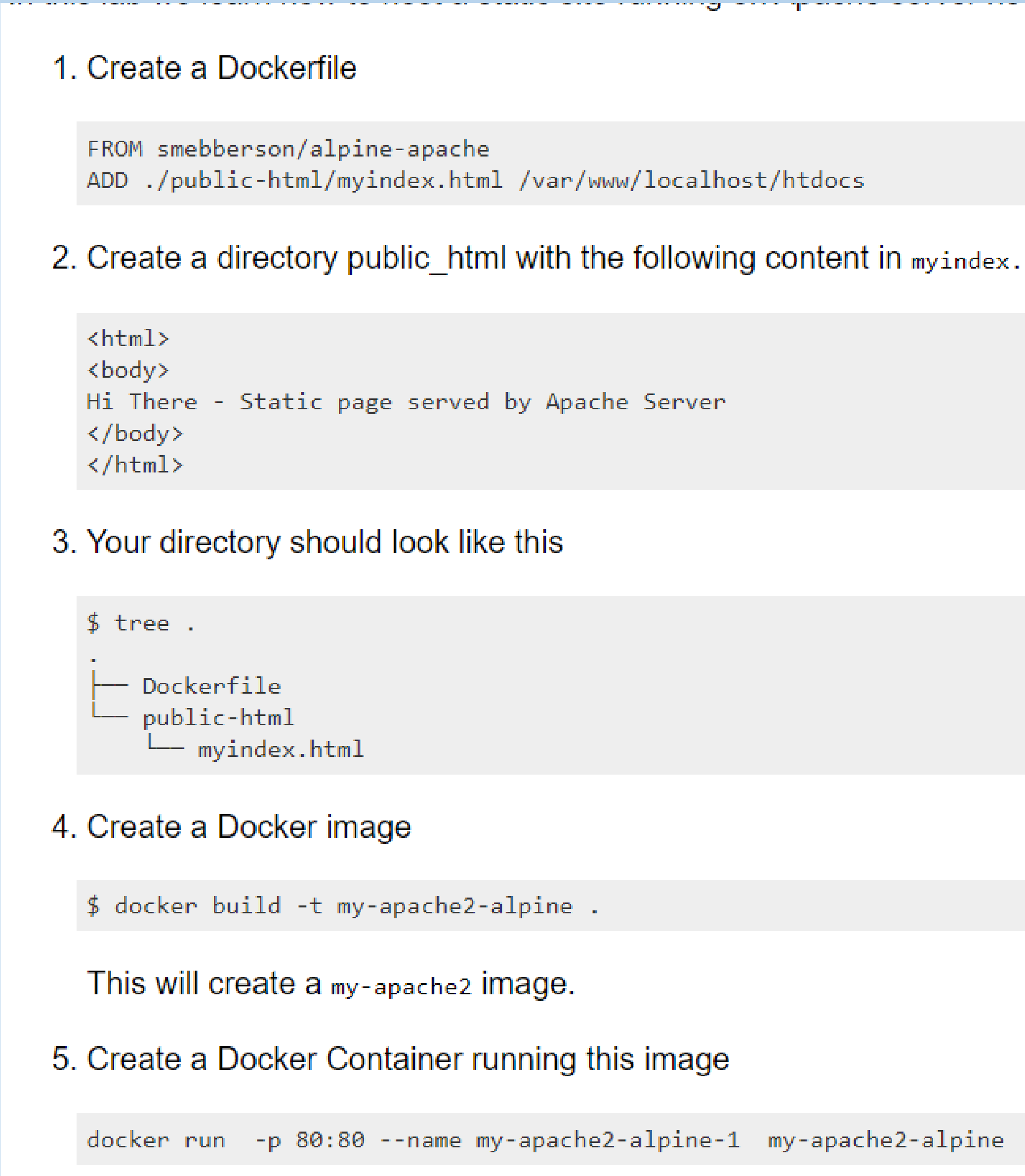
Check IP Address of the container

```
$ docker run alpine ifconfig
```

Launching a bash shell

```
$ docker run -i -t alpine /bin/bash
```

Docker Build Example



1. Create a Dockerfile

```
FROM smebberson/alpine-apache
ADD ./public-html/myindex.html /var/www/localhost/htdocs
```

2. Create a directory public_html with the following content in `myindex.`

```
<html>
<body>
Hi There - Static page served by Apache Server
</body>
</html>
```

3. Your directory should look like this

```
$ tree .
.
├── Dockerfile
└── public-html
    └── myindex.html
```

4. Create a Docker image

```
$ docker build -t my-apache2-alpine .
```

This will create a `my-apache2` image.

5. Create a Docker Container running this image

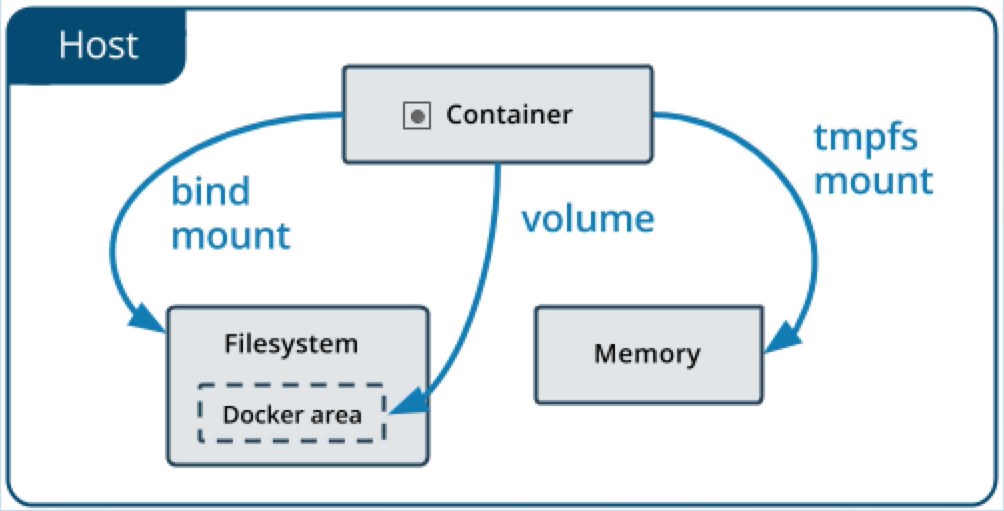
```
docker run  -p 80:80 --name my-apache2-alpine-1  my-apache2-alpine
```

Dockerfile

- Commands
  - FROM
  - ADD
  - RUN
  - COPY
  - EXPOSE
  - ENVIRONMENT

Volumes and Network Port Binding

- Mount Host Directory as a ‘volume’
  - Ugh overloaded term.. hear ‘volume’ means a rooted directory tree.

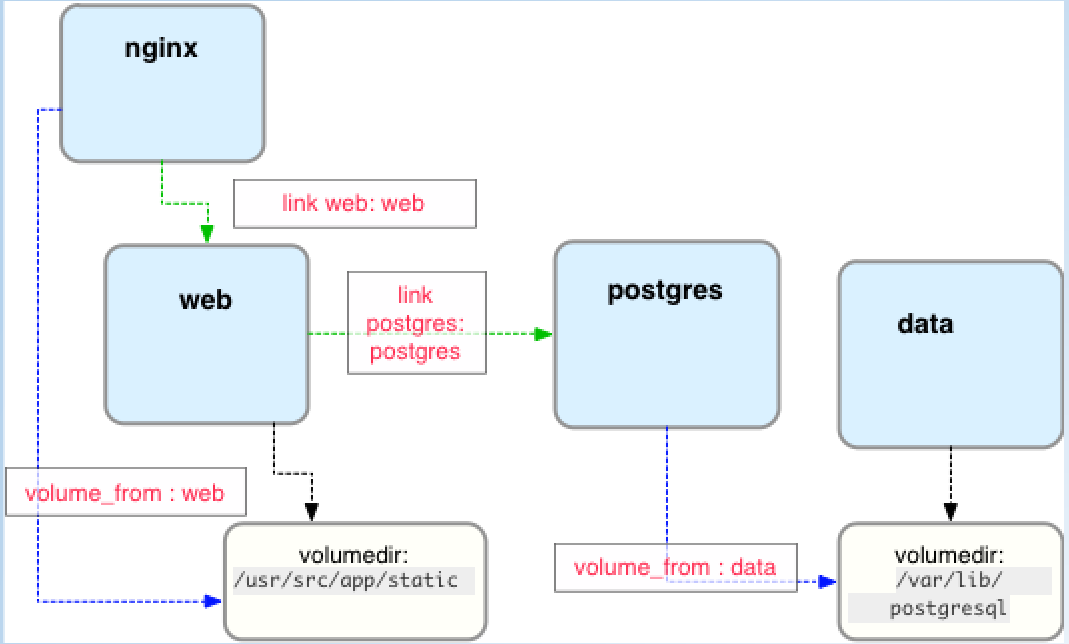


- Networking
  - Bind Host port to container port

Docker Compose

- http://containertutorials.com/linked/docker_compose.html

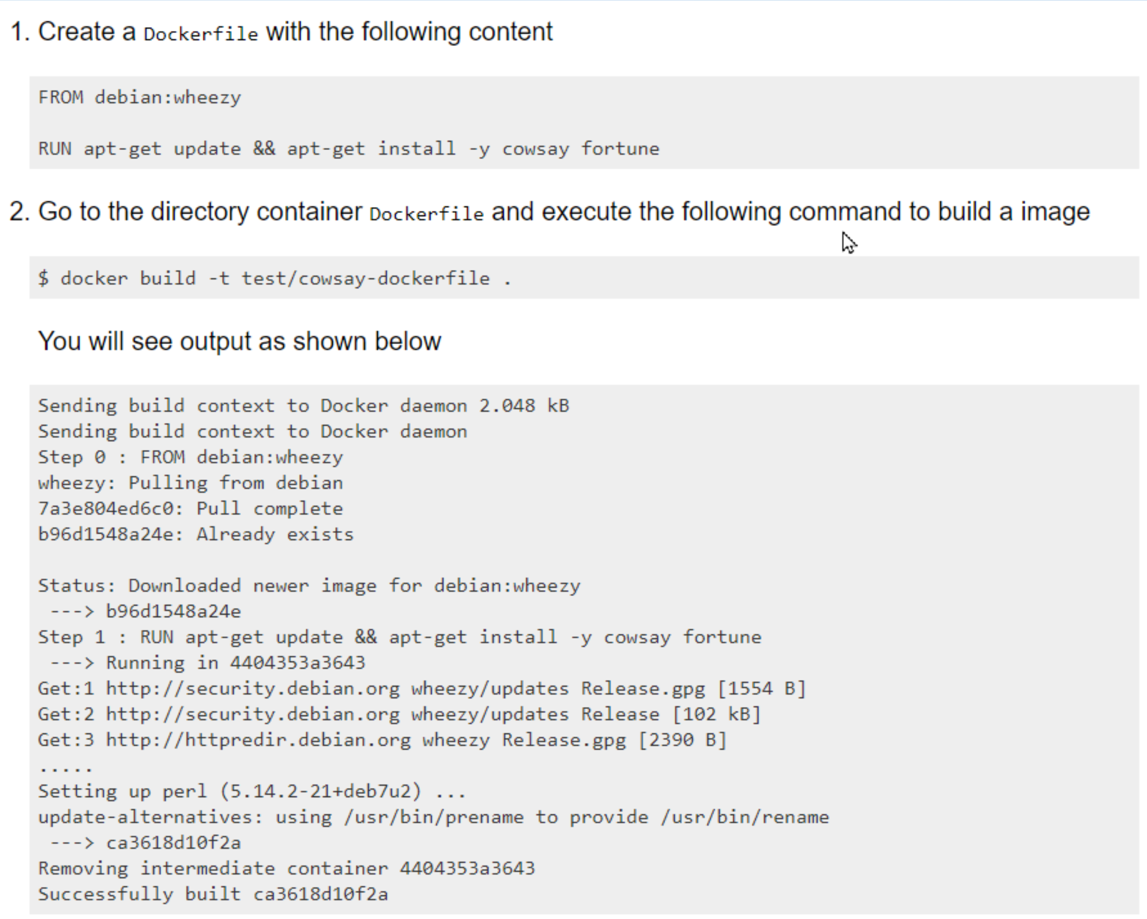
- Remember Flask servers are simple
    - Examples
    - Docker Compose with 2 ports/base



- http://containertutorials.com/docker-compose/spring-boot-app.html

Examples

- From



1. Create a `Dockerfile` with the following content

```
FROM debian:wheezy

RUN apt-get update && apt-get install -y cowsay fortune
```

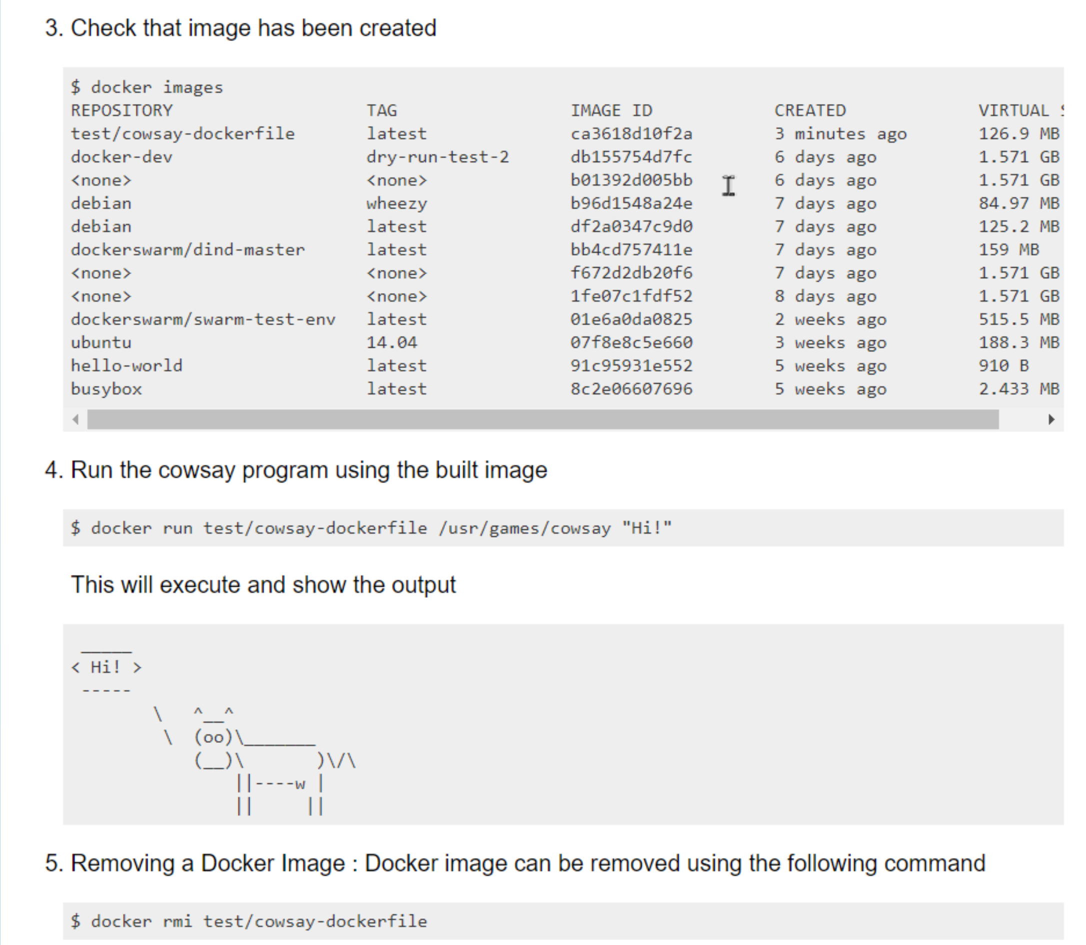
2. Go to the directory container `Dockerfile` and execute the following command to build a image

```
$ docker build -t test/cowsay-dockerfile .
```

You will see output as shown below

```
Sending build context to Docker daemon 2.048 kB
Sending build context to Docker daemon
Step 0 : FROM debian:wheezy
wheezy: Pulling from debian
7a3e804ed6c0: Pull complete
b96d1548a24e: Already exists

Status: Downloaded newer image for debian:wheezy
 ---> b96d1548a24e
Step 1 : RUN apt-get update && apt-get install -y cowsay fortune
 ---> Running in 4404353a3643
Get:1 http://security.debian.org wheezy/updates Release.gpg [1554 B]
Get:2 http://security.debian.org wheezy/updates Release [102 kB]
Get:3 http://httpredir.debian.org wheezy Release.gpg [2390 B]
.....
Setting up perl (5.14.2-21+deb7u2) ...
update-alternatives: using /usr/bin/prename to provide /usr/bin/rename
 ---> ca3618d10f2a
Removing intermediate container 4404353a3643
Successfully built ca3618d10f2a
```



3. Check that image has been created

```
$ docker images
REPOSITORY                  TAG              IMAGE ID        CREATED          VIRTUAL
test/cowsay-dockerfile      latest           ca3618d10f2a    3 minutes ago    126.9 MB
docker-dev                  dry-run-test-2   db155754d7fc    6 days ago       1.571 GB
<none>                      <none>           b01392d005bb    6 days ago       1.571 GB
debian                      wheezy           b96d1548a24e    7 days ago       84.97 MB
debian                      latest           df2a0347c9d0    7 days ago       125.2 MB
dockerswarm/dind-master     latest           bb4cd757411e    7 days ago       159 MB
<none>                      <none>           f672d2db20f6    7 days ago       1.571 GB
<none>                      <none>           1fe07c1fdf52    8 days ago       1.571 GB
dockerswarm/swarm-test-env  latest           01e6a0da0825    2 weeks ago      515.5 MB
ubuntu                      14.04            07f8e8c5e660    3 weeks ago      188.3 MB
hello-world                 latest           91c95931e552    5 weeks ago      910 B
busybox                     latest           8c2e06607696    5 weeks ago      2.433 MB
```

4. Run the cowsay program using the built image

```
$ docker run test/cowsay-dockerfile /usr/games/cowsay "Hi!"
```

This will execute and show the output

```
 _____
< Hi! >
 -----
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

5. Removing a Docker Image : Docker image can be removed using the following command

```
$ docker rmi test/cowsay-dockerfile
```

Docker Commands

- http://containertutorials.com/get_started
- docker <cmd>
  - info
    run
    run –it
    attach
    ps
    start
    stop
    logs
    rm
    pause/unpause
    network ls
    rename

Unravelling the Kubernetes Storage Abstraction

- There are Many Confusing Kubernetes Objects
  - Pod
  - Deployment, StatefulSet, DaemonSet
  - Volume
  - PersistentVolumeClaim
  - PersistentVolume
  - PersistentVolumeAttachment
  - StorageClass
  - CSIDriver

Where are the Containers?

- Pod
  - One or more Containers Running with a shared ‘localhost’

- Deployment
  - A Template for deploying several Pods over multiple nodes (interchangeably)

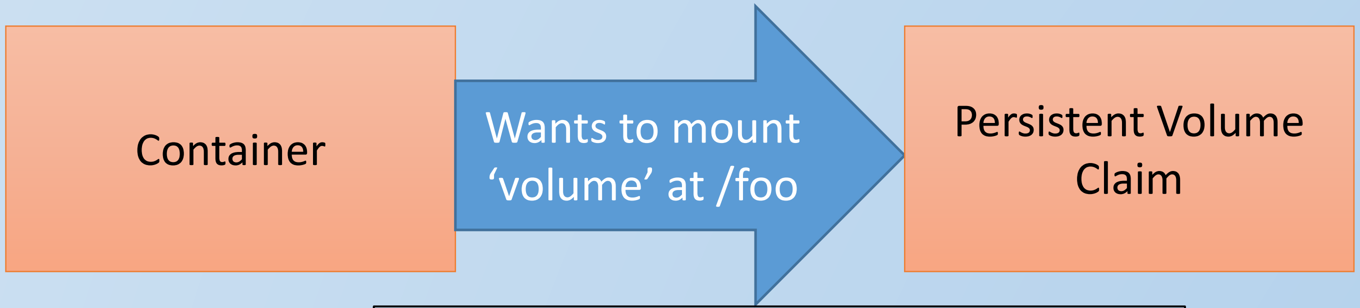
- StatefulSet
  - A Template for deploying several Pods over multiple nodes (sticky id)

- DaemonSet
  - Template for running several Pods one per Node (or subset of Nodes)

Persistent Storage Objects

- Volume
- PersistentVolumeClaim
- PersistentVolume
- PersistentVolumeAttachment
- StorageClass
- CSIDriver

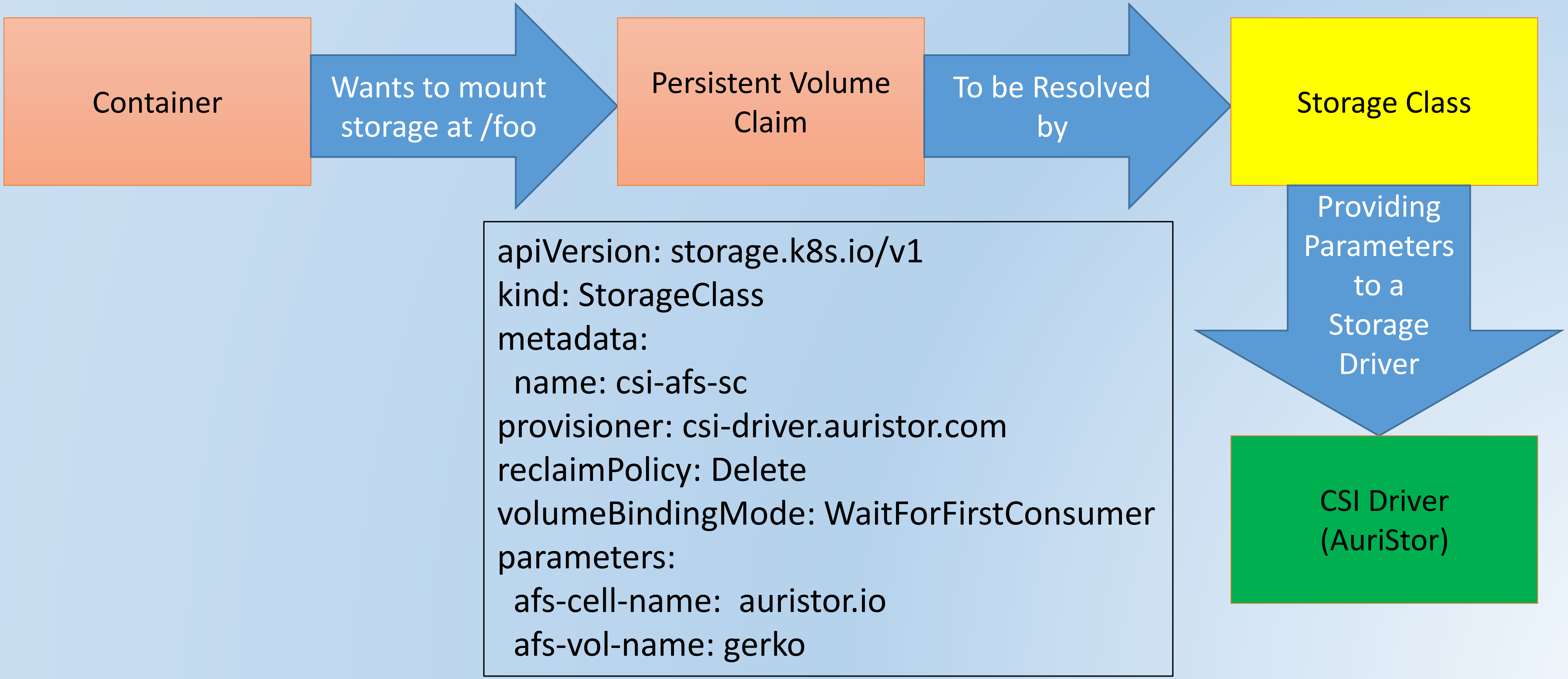
What the Application Developer Specifies



```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: csi-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-afs-sc
```

```
kind: Pod
apiVersion: v1
metadata:
  name: my-csi-app
spec:
    containers:
    - name: my-frontend
      image: busybox
      volumeMounts:
      - mountPath: "/data"
        name: my-csi-volume
      command: [ "sleep", "1000000"
]
  volumes:
    - name: my-csi-volume
      persistentVolumeClaim:
        claimName: csi-pvc
```

What the Cluster (Storage) admin Specifies



Container

Wants to mount storage at /foo

Persistent Volume Claim

To be Resolved by

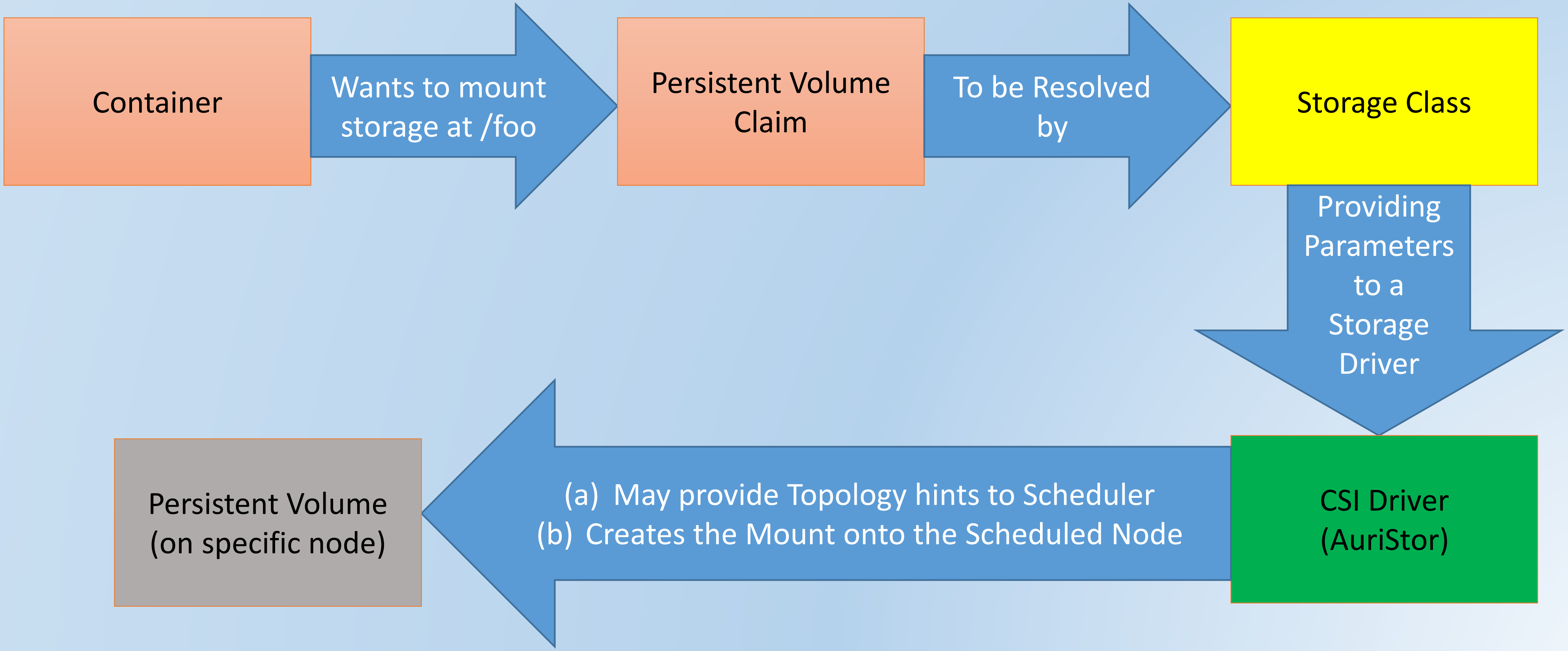
Storage Class

Providing Parameters to a Storage Driver

CSI Driver (AuriStor)

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-afs-sc
provisioner: csi-driver.auristor.com
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer
parameters:
  afs-cell-name:  auristor.io
  afs-vol-name: gerko
```

What the Kubernetes CSI Driver Does



Container

Wants to mount storage at /foo

Persistent Volume Claim

To be Resolved by

Storage Class

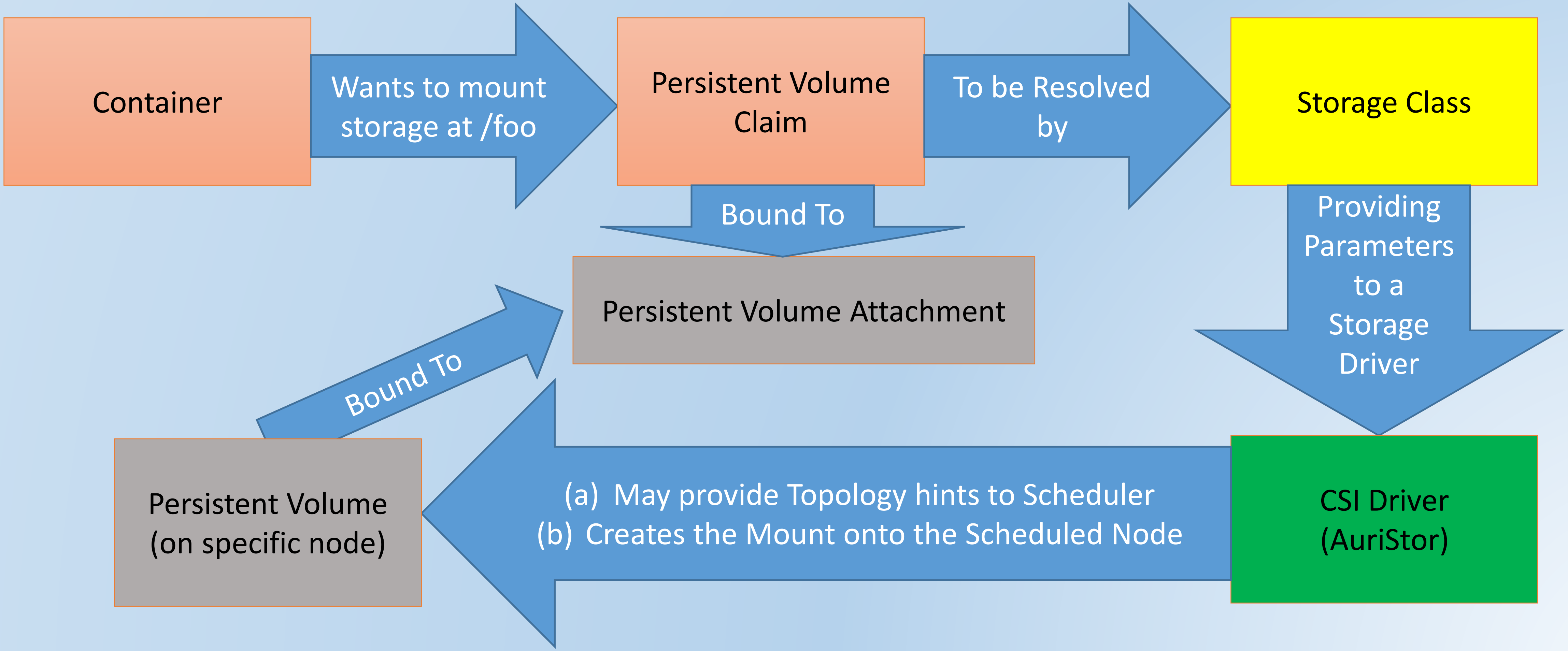
Providing Parameters to a Storage Driver

CSI Driver (AuriStor)

(a) May provide Topology hints to Scheduler
(b) Creates the Mount onto the Scheduled Node

Persistent Volume (on specific node)

What the external-attacher does



Container

Wants to mount storage at /foo

Persistent Volume Claim

To be Resolved by

Storage Class

Bound To

Persistent Volume Attachment

Providing Parameters to a Storage Driver

Bound To

Persistent Volume (on specific node)

(a) May provide Topology hints to Scheduler
(b) Creates the Mount onto the Scheduled Node

CSI Driver (AuriStor)

The Storage Class Parameters to CSI Driver

- AuriStor Driver Specific
  - Volume:
    - Existing Cell + Volume
    - Scratch + Quota
  - Topology
    - Don’t care where the volume is
    - Schedule ‘near’ a File Server (ie Rack) with Volume (Replicate if necessary)
      - Do not Start until replication competes
      - Start immediately if none, but also start replication

Distributed Systems Study Group

First Meeting: Dec 3rd, 2019

Questions?

Gerry Seidman
gerry@iamx.com