

The Aerospike logo, consisting of the word "AEROSPIKE" in a sans-serif font, enclosed within a thin rectangular border.

AEROSPIKE

An abstract white wireframe graphic on the left side of the slide, resembling a stylized globe or a complex network structure, set against a red background with a cityscape pattern.

Multi-Host, Multi-Network Persistent Containers

Powering New Opportunities at Scale

Conclusion

- **Containers + Databases = Happy Developers**
- **Ephemeral Containers + Databases = DevOps headaches**
- **4 Things you must use to evaluate**
 - Data Redundancy
 - Dynamic Self Discovery & Cluster formation
 - Self Healing (as containers enter and leave)
 - Application Tier discovery of Database Cluster

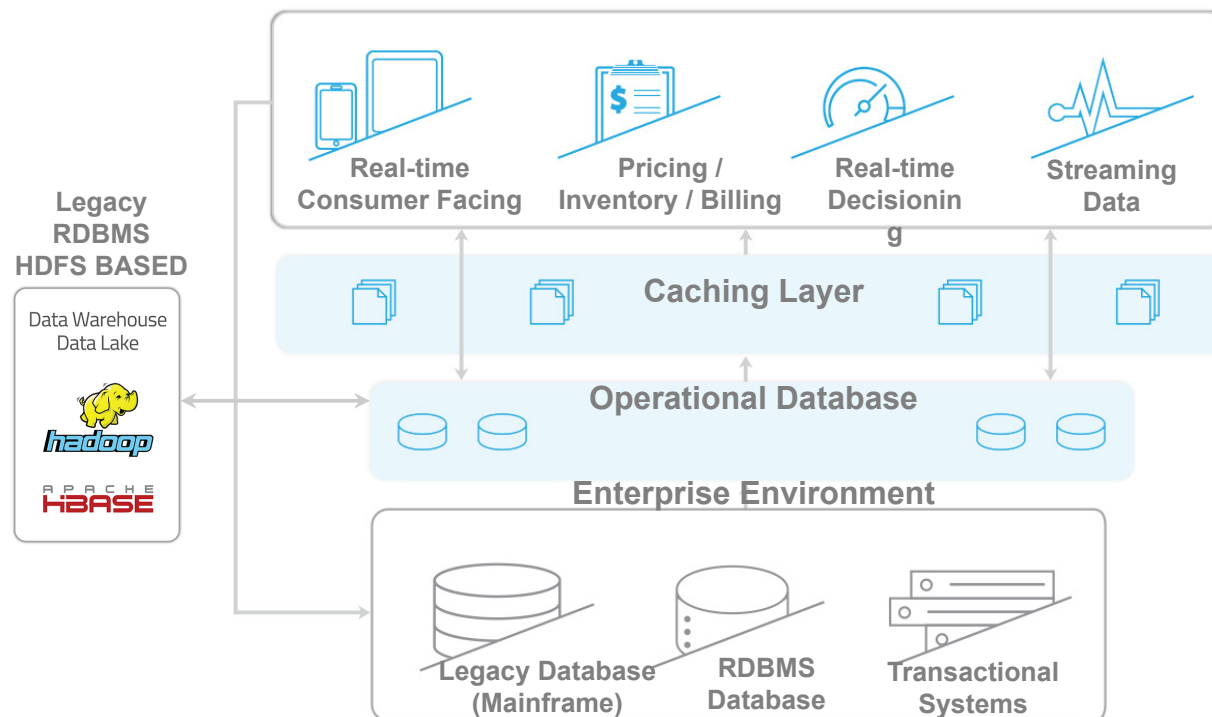


Part One

*"Here's another nice mess you have
got me into"*

Laurel & Hardy circa 1929

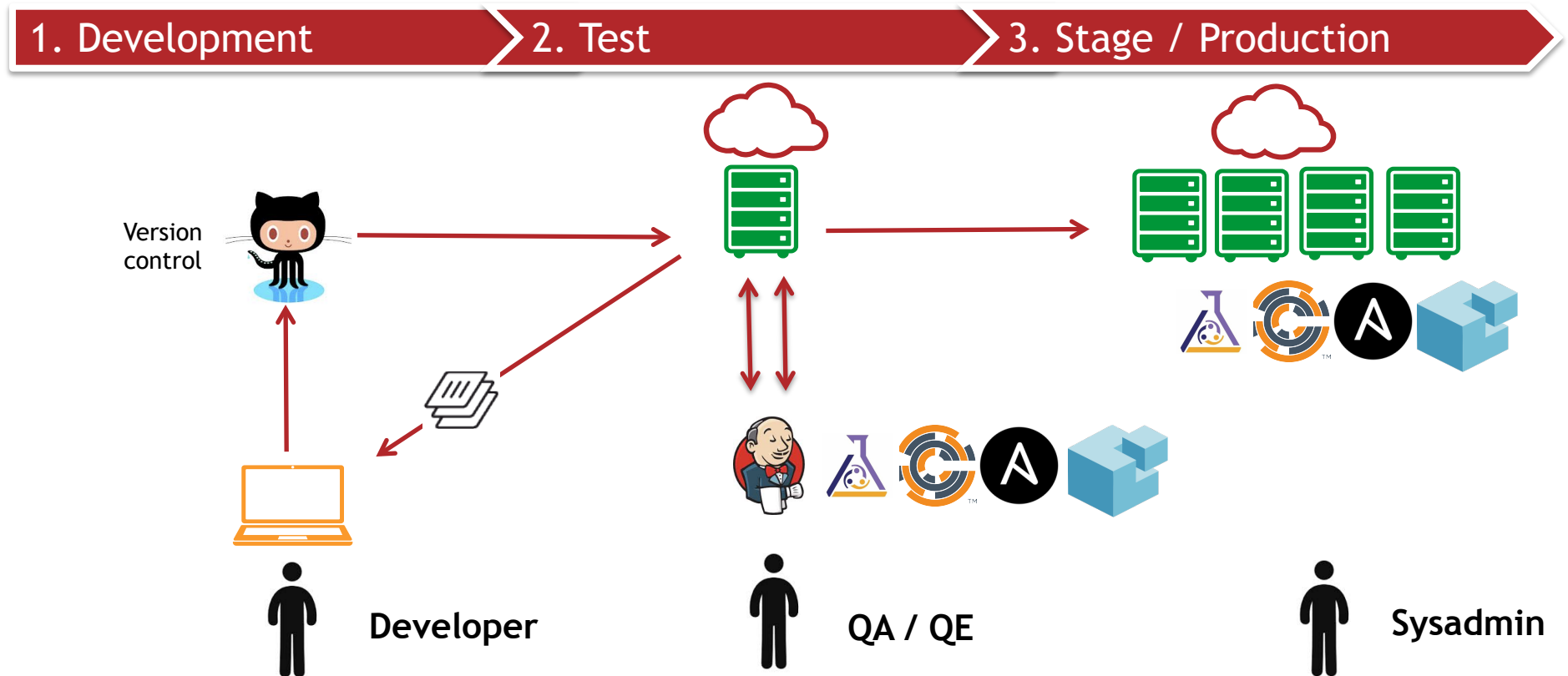
Existing Architectures Are Broken



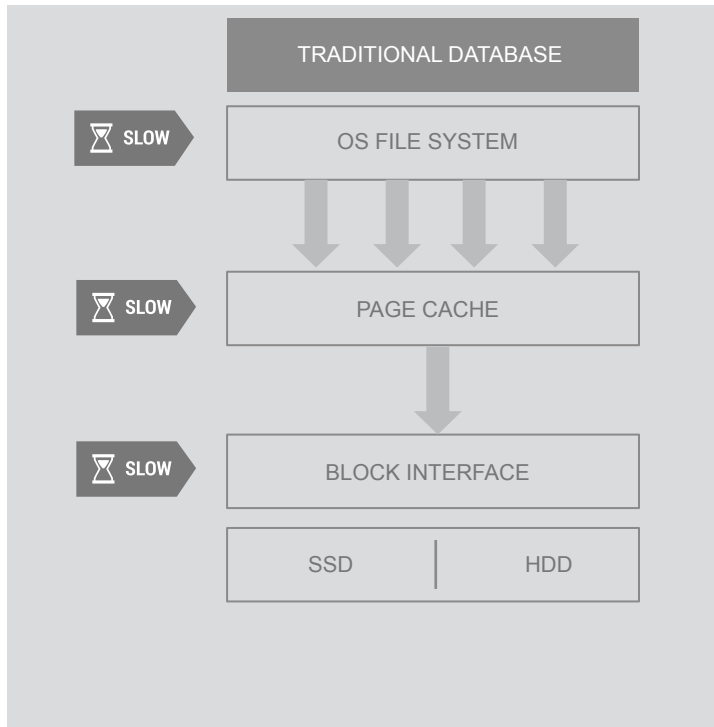
Challenges

- Complex
- Maintainability
- Durability
- Consistency
- Scalability
- Cost (\$)
- Data Lag

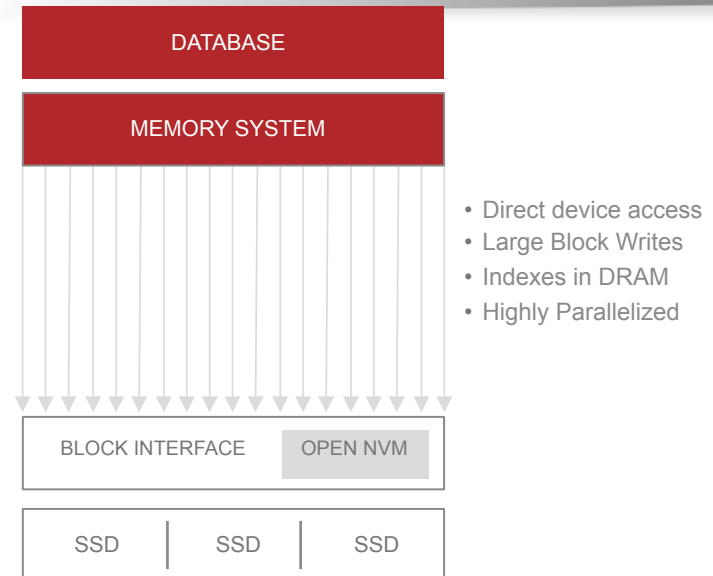
Existing Deployment Models Are Broken



Infrastructure Cannot be Fully Utilized



What You Have



What You Want

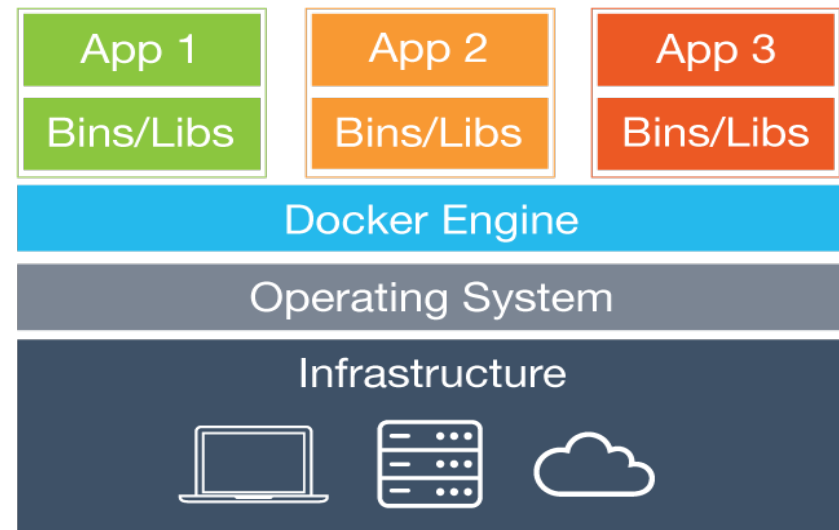
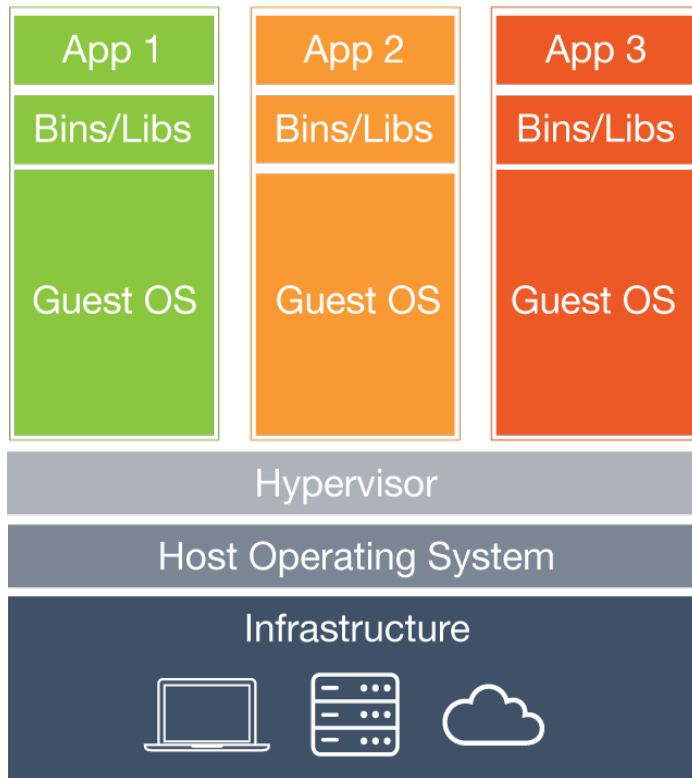
The background is a solid red color with a series of thin, curved, and overlapping lines that create a sense of motion and depth, resembling a stylized tunnel or a flowing liquid. The lines are lighter in color where they are more prominent and fade into the red background as they recede.

Part Two Containers

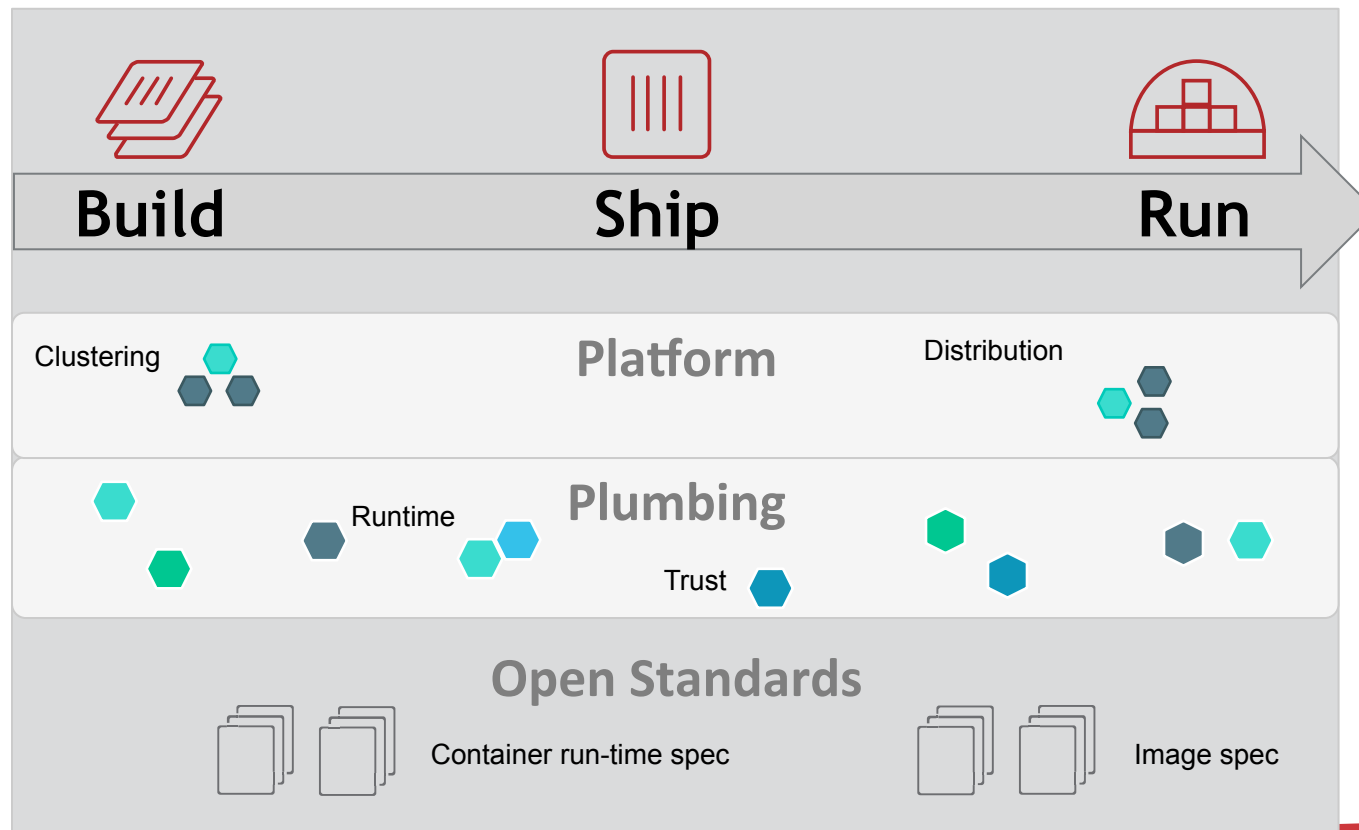
What do Containers give me?

- **Encapsulation of Dependencies**
 - O/S packages & Patches
 - Execution environment (e.g. Python 2.7)
 - Application Code & Dependencies
- **Process Isolation**
 - Isolate the process from anything else running
- **Faster, Lightweight virtualization**

Containers vs. Virtual machines



Container Mission – Reduce Complexity



Dockerfile - Example

```
FROM python:2.7
ADD . /code
WORKDIR /code
RUN apt-get update
RUN apt-get -y install python-dev
RUN apt-get -y install libssl-dev
RUN pip install --no-cache-dir -r requirements.txt
EXPOSE 5000
CMD python app.py
```

Open Container Initiative (OCI) – Polyglot Vendors

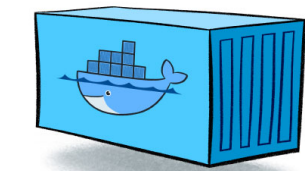
Coalition of industry leaders join forces to eliminate fragmentation

- Form a vendor-neutral, open source governance model under the Linux Foundation
- Establish common standards for container format and runtime
- Docker donated its container format, runtime and associated specifications <http://www.opencontainers.org/>



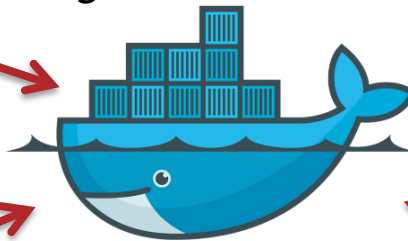
Docker Landscape in Pictures

*Containers encapsulates
your code, dependencies...*



Container

*Containers are run
by Docker Engine*



Docker Engine



Docker Machine

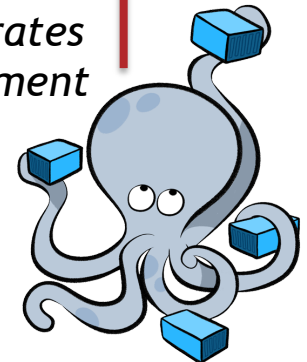
*Machine provisions
Docker Engines*

*Swarm clusters
Docker Engines*



Docker Swarm

*Compose orchestrates
Container deployment*



Docker Compose

The background of the slide is a solid red color with a series of thin, curved, light-colored lines that create a sense of motion and depth, resembling a tunnel or a fast-moving path.

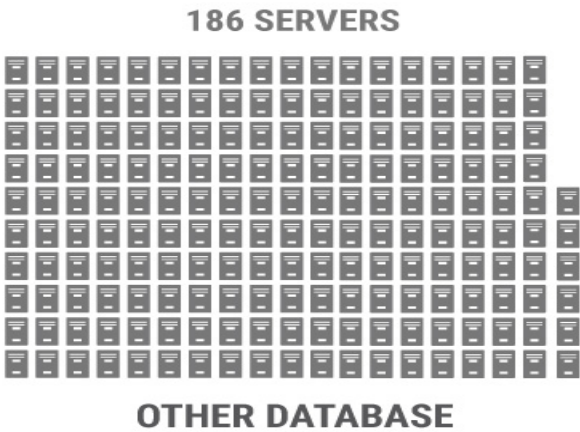
Part Three – Aerospike The Enterprise NoSQL Database

The Bottom Line

10x FASTER OR 10x FEWER

**ACTUAL
CUSTOMER
REQUIREMENTS**

99% < 1ms
500K TPS
10TB Storage
2x Replication



Indexes Location	RAM	RAM
Data Location	RAM	SSD
Persistence	Hard Drive	SSD
Storage per server	180 GB (on 196 GB server)	2.8 TB (4 x 700 GB)
Cost per server	\$8,000	\$11,000
Server costs	\$1,488,000	\$154,000
Power/server	0.9 kW	1.1 kW
Power (2 years) \$0.12 per kWh ave. US	\$352,200	\$32,400
Maintenance(2 years) \$3600/server	\$670,000	\$5042
Total	\$2,510,000	\$236,800

Built for Flash

Utilization



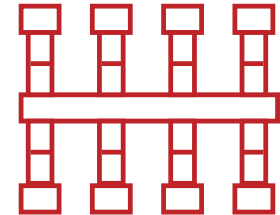
- Primary Key index in parentless Red-Black tree in DRAM
 - Data in DRAM or SSD
 - Secondary B-Tree indexes in SSD
- Proprietary Log Structured File system
- Parallelize reads/writes to multiple SSDs

Fundamental IP



- Self-managing nodes of a distributed database cluster
- Cluster-node load balancing in a distributed system
- Hybrid DRAM-SSD memory system
- Real-time transaction scheduling

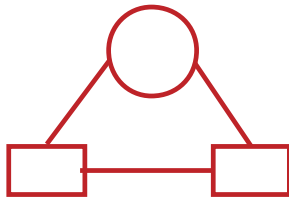
X Point



- Still requires specific optimizations
- High IOP and durability NVMe required by customers
- Excellent working relationship with Intel for Xpoint on DDR

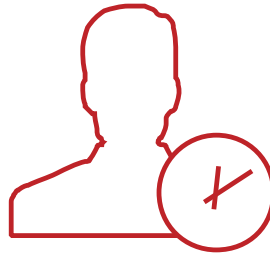
Master-Based Clustering

Architecturally Correct



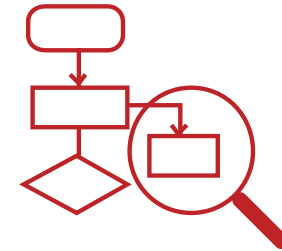
- Hybrid Peer-to-Peer with Master
- Provides either Availability or Consistency
- Mastering is required for transaction correctness
- Sync writes within a Cluster

High Availability



- Aerospike's HA reputation is unmatched
- Cross Data Center replication (XDR) for HA/DR
- FinServ and Telecom customers using application-level hot standby
- Extends to Conflict Resolution

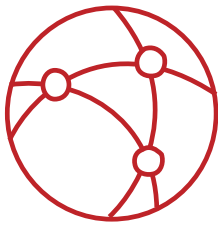
High Consistency



- Fits the architecture
- Demanded by Enterprise customers

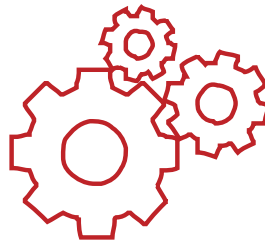
Developer Experience

Rich & Simple



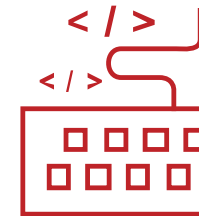
- Schema free
- Geospatial
- List & Map server-side manipulation
- Secondary Indexes

Integration



- Frameworks
 - SpringData
 - Play
- Connectors
 - JDBC
 - Spark / Hadoop

Deployment



- Docker integration
- Orchestration (Mesos, Kubernetes) – in progress
- IPv6 - in progress
- Security with transport encryption, certificate based authentication (in progress)

Real-time Fraud Prevention

Challenge

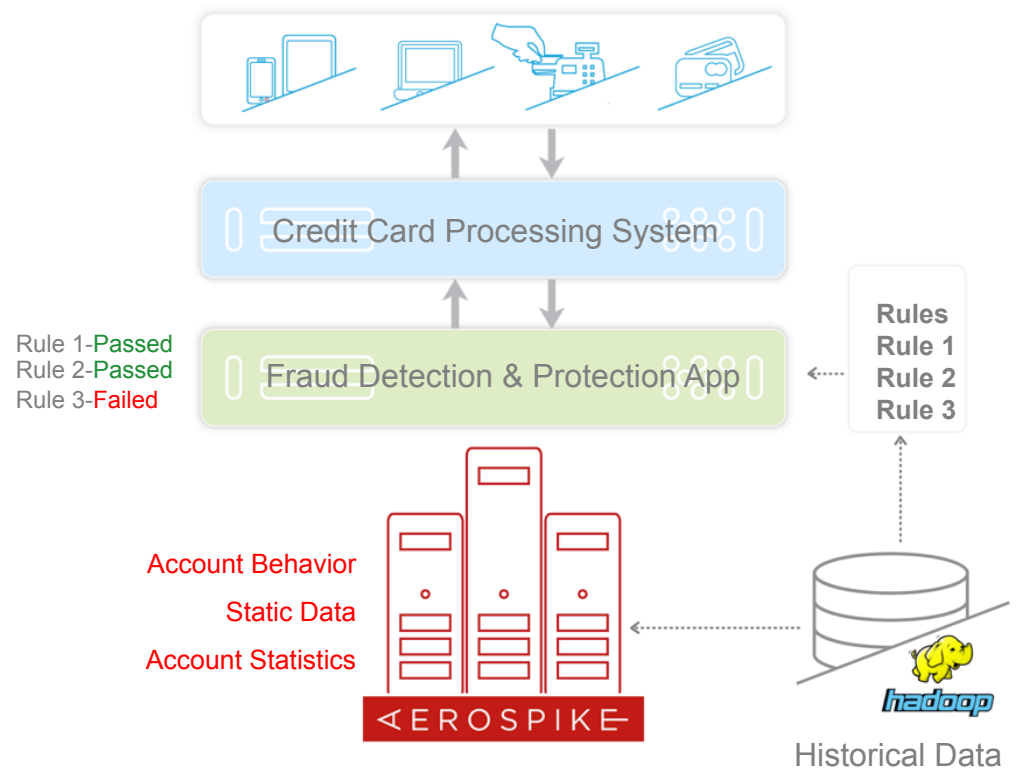
- Overall SLA 750 ms
- Loss of business due to latency
- Every credit card transaction requires hundreds of DB reads/writes

Need to Scale Reliably

- 10 → 100 TB
- 10B → 100 B objects
- 200k → 1 Million+ TPS

Aerospike In-Memory NoSQL

- Built for Flash
- Predictable low latency at high throughput
- Immediate consistency, no data loss
- Cross data center (XDR) support
- 20 server cluster
- Dell 730xd w/ 4NVM SSDs

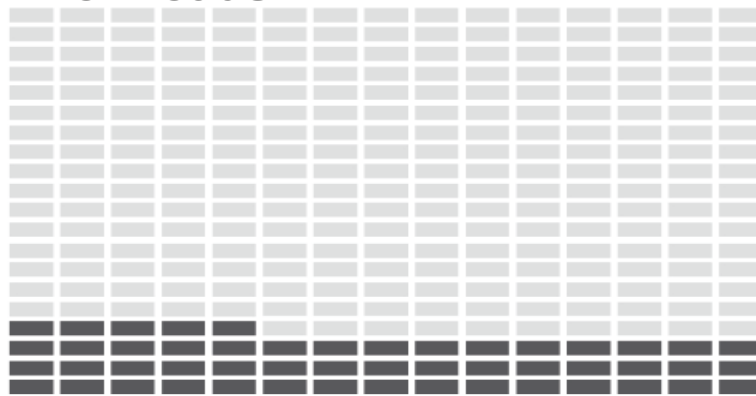


Cloud Deployment: 1 Million Writes/Sec on Google Compute

- Aerospike hits 1M writes/sec with 6x fewer servers than Cassandra
- Delivers consistent low latency with no jitter for both read and write workloads

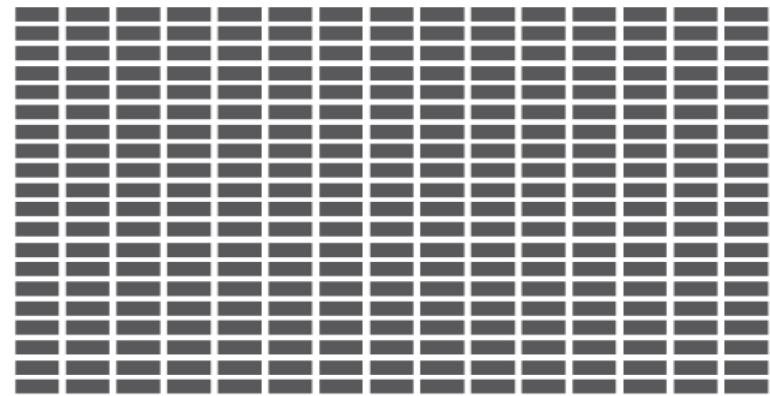


Google Cloud Platform



AEROSPIKE

50 nodes



Cassandra

300 nodes

New results: 20 nodes, and 4M reads per second

The background of the slide is a solid red color with a series of smooth, flowing, wavy lines that create a sense of motion and depth, resembling a stylized landscape or a tunnel effect.

Part Four

Databases and Docker

Requirements

- **Data Redundancy**
 - Containers are Ephemeral – Need more than one copy of the data
- **Dynamic Self Discovery & Cluster formation**
 - Need to start and stop Containers when needed
 - Clusters needs to grow and shrink dynamically
- **Self Healing**
 - Loss of nodes must not be fatal to the cluster integrity
 - Addition of nodes must scale capacity
- **Application Tier discovery of Database Cluster**
 - Automatic discovery of nodes
 - Automatic routing of requests to the correct nodes

Example: Aerospike and Docker

- **Data Redundancy**
 - Automatic Replication of Data to "n" nodes
- **Dynamic Self Discovery & Cluster Formation**
 - Shared nothing architecture – all nodes equal
 - Multi-cast & Mesh Networking models
- **Self Healing**
 - Automatic hashing of keys across the cluster & rebalancing
 - RIPEMD-160 collision free algorithm with Smart Partitions™
- **Application Tier discovery of Database Cluster**
 - Automated cluster discovery with Smart Client™
 - Java, C/C++, C#, Python, Node.js

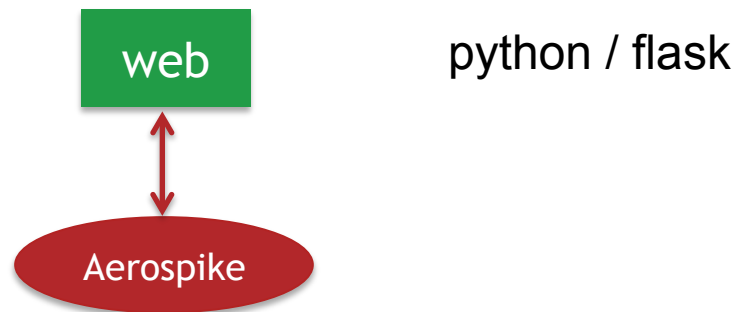
The background is a solid red color with a series of thin, curved, and overlapping lines that create a sense of motion and depth, resembling a stylized road or a fluid surface. The lines are lighter in color where they are more prominent and fade into the red background as they recede.

Part Five Demo

Demo: Development through to Production

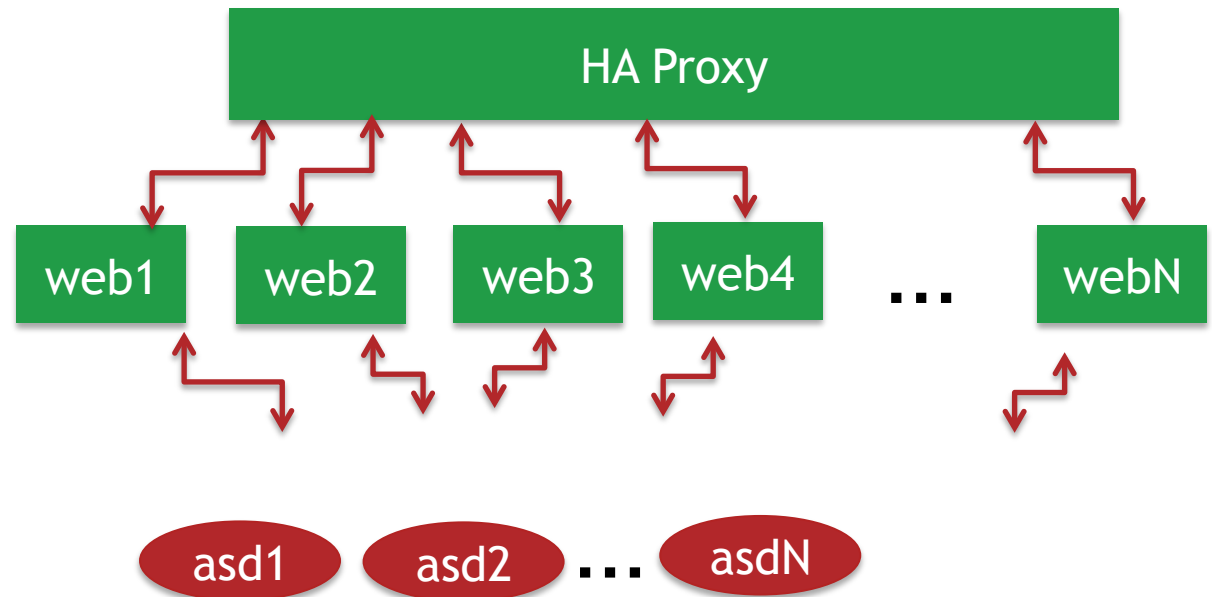
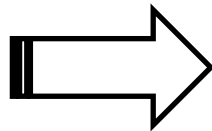
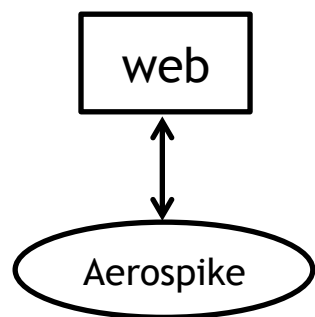
- **Build & Run an App in Development**
 - Python + Aerospike
- **Deploy to a Swarm cluster in Production**
 - Add more Web containers behind HAProxy
- **Scale Aerospike Cluster in production**
 - Add more Database nodes

Lets build an App!



Development

Scale in Production



Development

Production

Demo 1 : Build an App

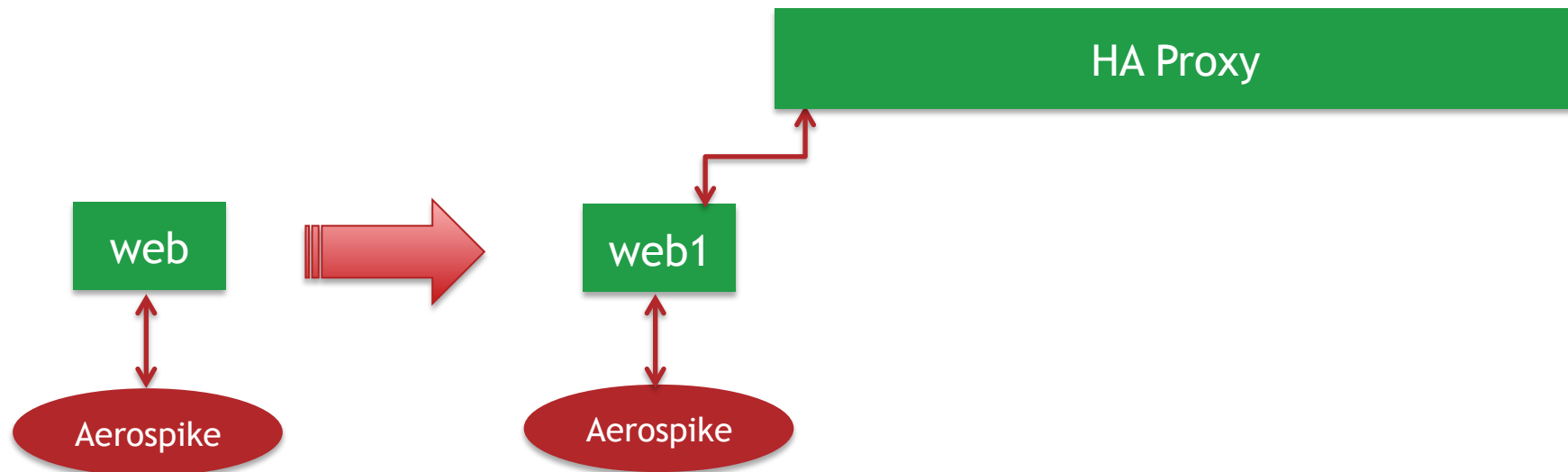
Dockerfile

```
FROM python:2.7
ADD . /code
WORKDIR /code
RUN apt-get update
RUN apt-get -y install python-dev
RUN apt-get -y install libssl-dev
RUN pip install --no-cache-dir -r requirements.txt
EXPOSE 5000
CMD python app.py
```

docker-compose.yml

```
web:
  build: .
  ports:
    - "5000:5000"
  links:
    - aerospike
  hostname: dev.awesome-counter.com
  environment:
    - AEROSPIKE_HOST=dev_aerospike_1
aerospike:
  image: aerospike/aerospike-server:latest
  volumes:
    - $PWD:/etc/aerospike
```

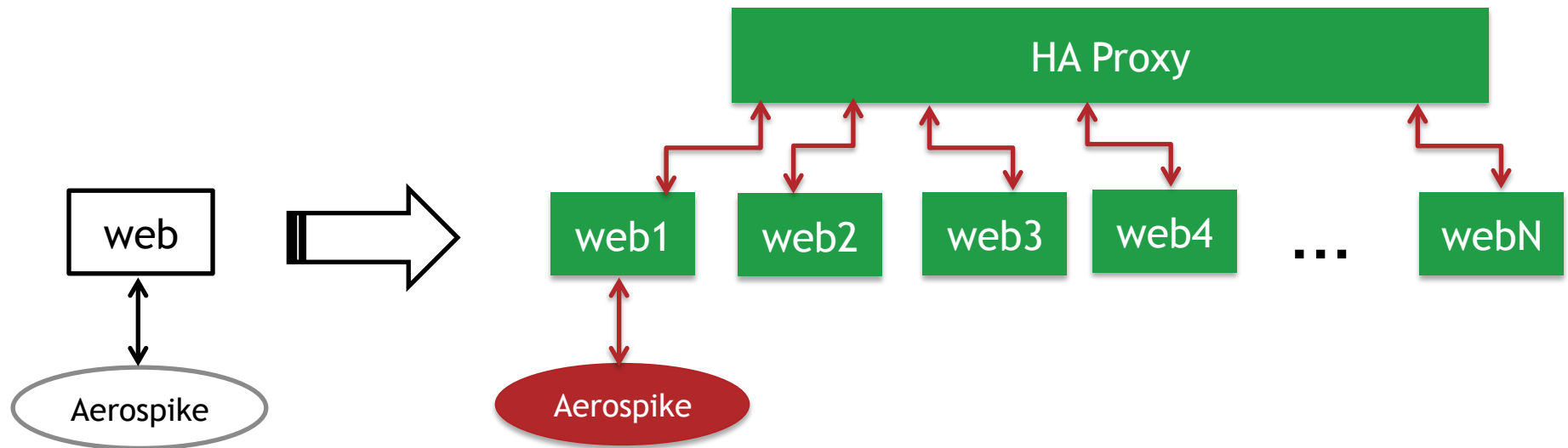
Roll the App to Production behind HA Proxy



Development

Production

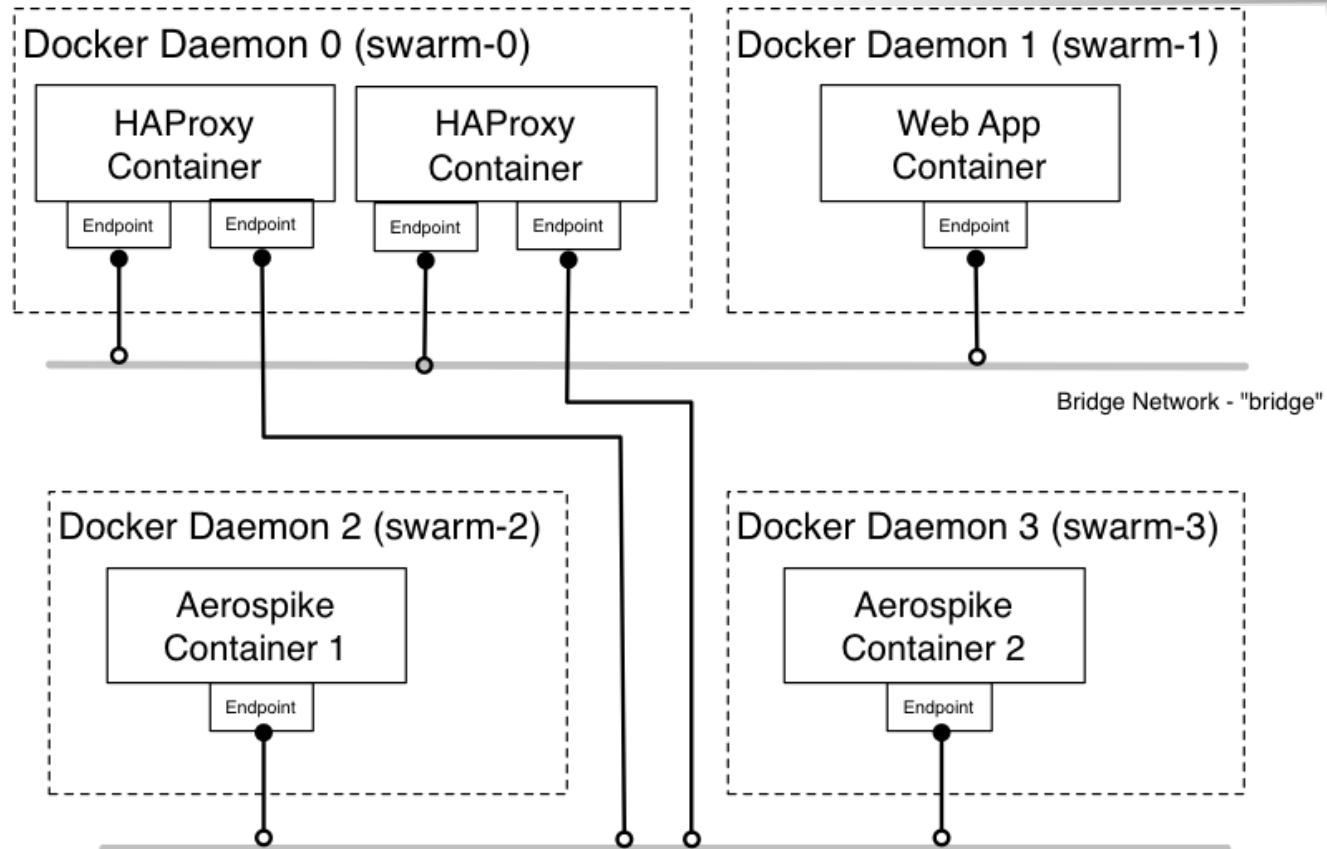
Scale the web tier



Development

Production

Docker Networking



Demo 2 : Scale the Web Tier

aes_base_cluster.yml

discovery:

image: aerospike/interlock:latest

environment:

- "DOCKER_HOST"

volumes:

- "/var/lib/boot2docker:/etc/docker"

command: "... --plugin aerospike start"

aerospike:

image: aerospike/aerospike-server:latest

volumes:

- "\$PWD:/etc/aerospike"

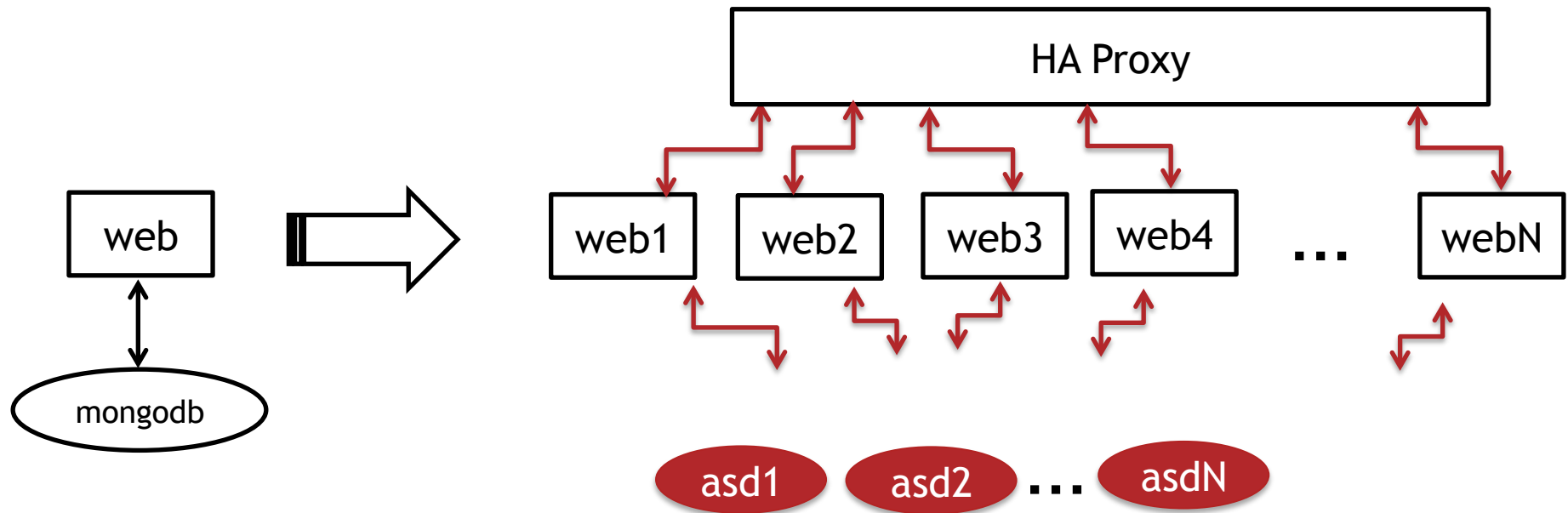
docker-compose.yml

```
haproxy:
  extends:
    file: haproxy.yml
    service: haproxy-server
  environment:
    - "constraint:node==swarm-0"
  net: bridge

web:
  image: alvinr/demo-webapp-as:latest
  extends:
    file: haproxy.yml
    service: haproxy-app
  environment:
    - AEROSPIKE_HOST=prod_aerospike_1
  net: prod
```

```
aerospike:
  extends:
    file: aes_base_cluster.yml
    service: aerospike
  image: aerospike/aerospike-server:3.7.1
  labels:
    - "com.aerospike.cluster=awesome-counter"
  environment:
    - "affinity:com.aerospike.cluster!=awesome-counter"
  net: prod
```

Scale the Aerospike cluster



Development

Production

Demo 3 : Scale the Cluster

Docker Event API & Interlock

■ API for Docker Events

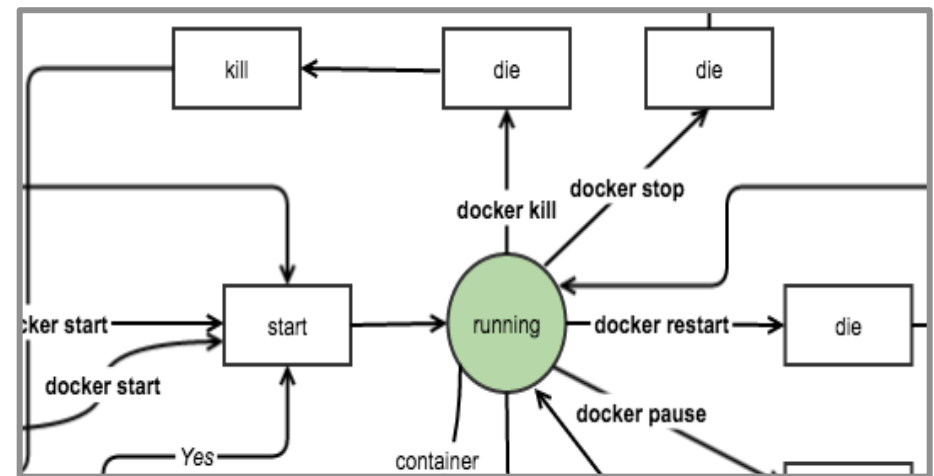
- Start / Stop / Die etc
- https://docs.docker.com/engine/reference/api/docker_remote_api/

■ Interlock – Evan Hazlett

- Framework to listen and publish events
- Plugin Framework (e.g. HAPROXY)
- <https://github.com/ehazlett/interlock>

■ Aerospike Interlock plugin

- Add / Remove node from Cluster
- <https://github.com/aerospike/interlock>



Interlock Plugin - Aerospike

- <https://github.com/aerospike/interlock>

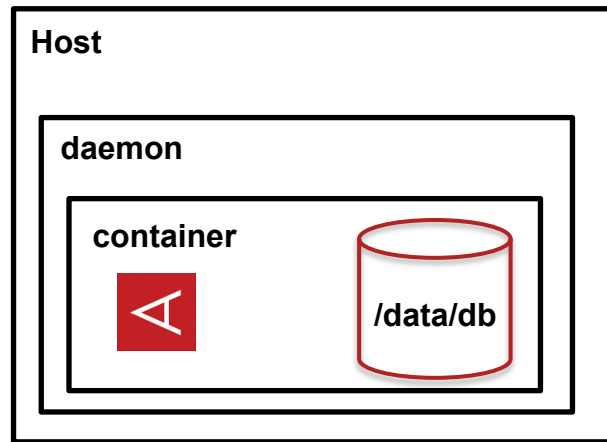
```
func (p AerospikePlugin) runAsinfoTip(args ...string) bool{
    asinfo, err := exec.LookPath("asinfo")
    if err != nil{
        log.Errorf("error finding asinfo binary: %s", err)
        return false
    }
    time.Sleep(time.Second*5)    //sleep 5s for ASD to be ready
    cmd := exec.Command(asinfo,args...)
```

The background of the slide is a solid red color with a series of thin, curved, light-colored lines that create a sense of motion and depth, resembling a stylized road or a tunnel. The lines are more prominent on the right side and fade towards the left.

Part Six

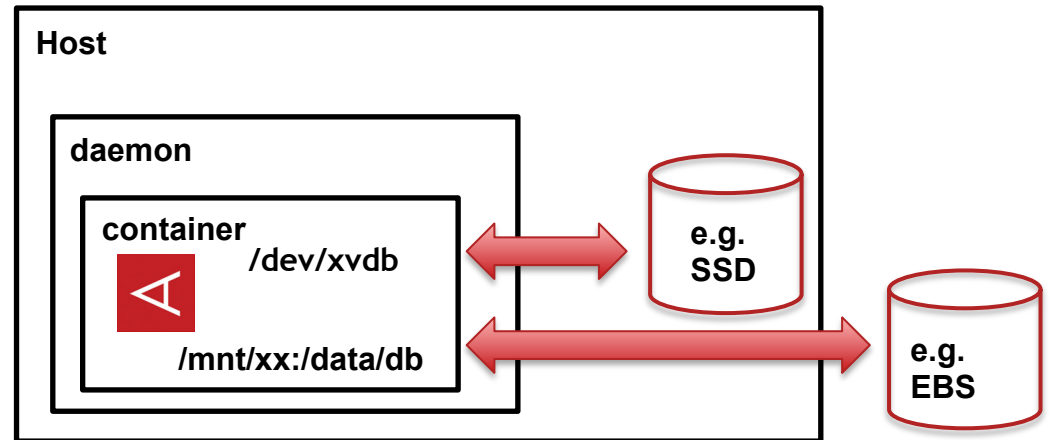
Considerations & Conclusion

Storage: Inside or outside the container?



Inside

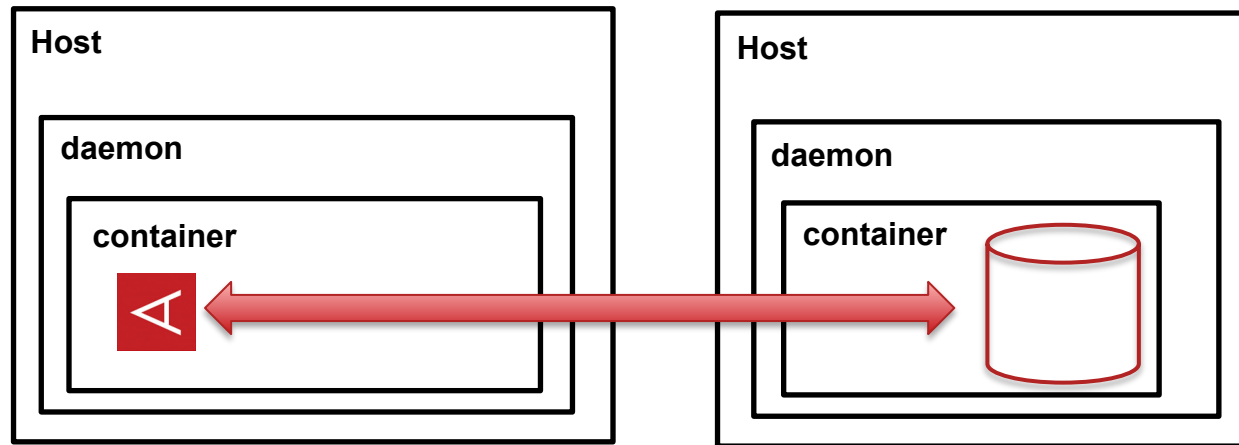
- Encapsulation of Concerns



Outside

- Separation of Concerns
- Storage Features (e.g. Snapshots)

Storage: Data Container?



Data Container

- `--volumes-from <container name>`
- Managed like other containers
- Special rules for Destruction
- TBD: Performance

Summary

One solution from Dev -> Production

- Define Container, their contents and how they work together once
- Deploy the same images in Dev, Pre-Prod and Production across Platforms

Running Docker & Database in Production

- Ops define the whitelisted images, security policies etc.
- Dev use approved images to build upon
- Eliminate the complexity (and cost) of deployment
- Scale up & down in a Flexible and Simple way

Thanks and Q&A

- **Code**

- <http://github.com/alvinr/docker-demo/tree/master/aerospike>

- **Docker Images**

- <http://hub.docker.com/r/aerospike/>

- **Aerospike & Docker deployment guide**

- http://www.aerospike.com/docs/deploy_guides/docker/

- **Contact me!**

- alvin@aerospike.com
- [@jonnyeight](#)