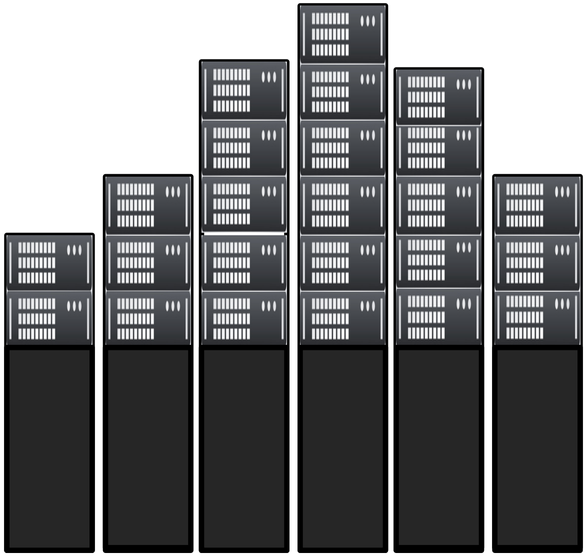


Lessons learned from running heterogeneous workload on Mesos

Imran Shaikh
Lead/Architect
yp

Blog http://elasticcompute.io
@imranshaikh

MesosCon 2016

Agenda

- Heterogeneous workload
- Isolation techniques
- Heterogeneous workload problems:
  - Structured logging
  - Application secrets
  - Application config management
  - Running databases
  - Integrations with existing infrastructure
  - Isolating resource hogs
  - etc.
- Other production tweaks
- Our solution
- What should be optimal solution?
- Conclusion
- Q/A

Heterogeneous workload

- Highly critical billing systems
- Traditional Java Apps
- Web based Apps
- Infrastructure tools
- Batch processing
- Message queues
- Map reduce jobs
- Build pipeline jobs
- NoSQL databases
- Relational databases

Heterogeneous workload

- Java
- Python
- Node.js
- Go Lang
- Ruby/Rails
- C/C++
- Scala
- JavaScript
- Perl
- PHP

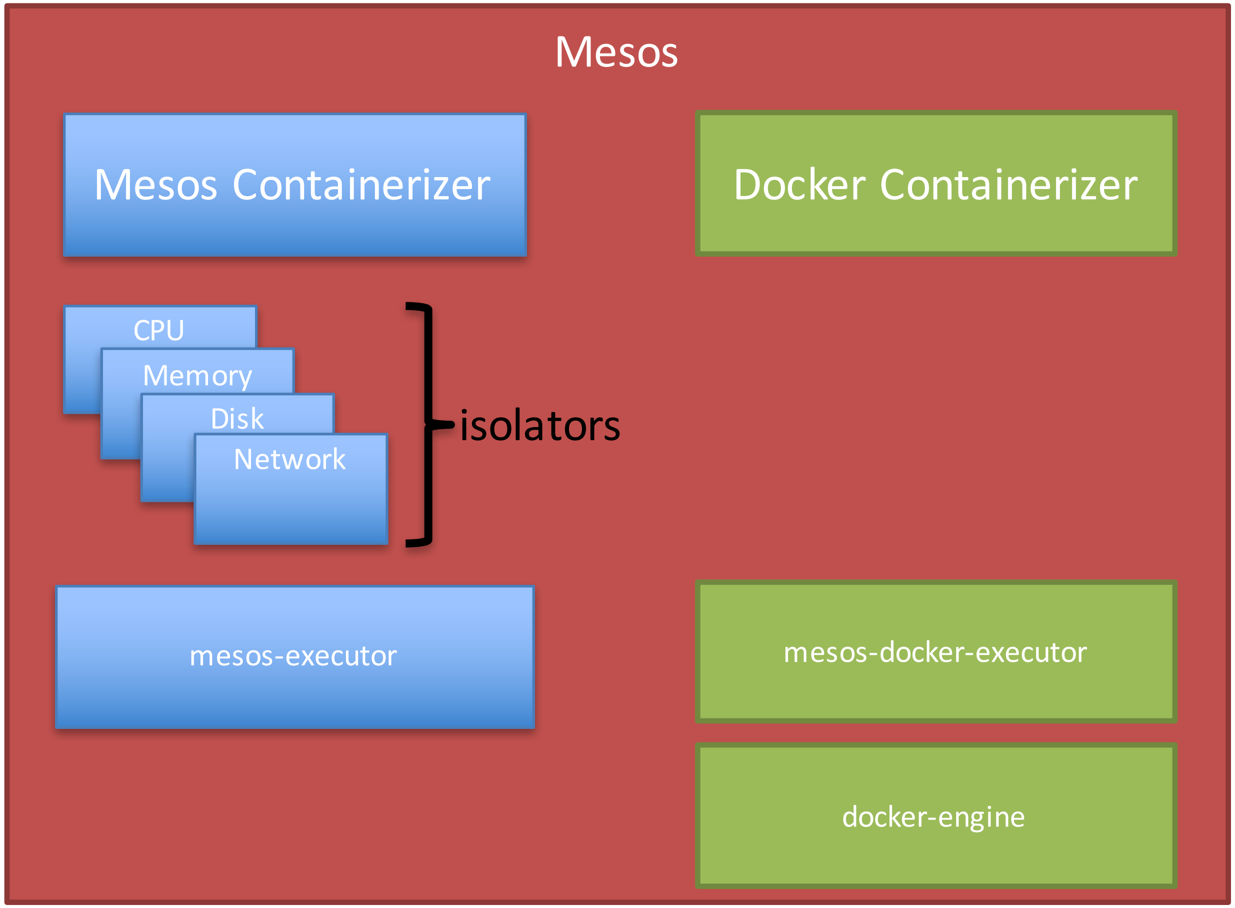
Why is it challenging?

- Workload often show diversity in terms of resource requirements, priority and performance objectives
- Some of the workload requires resource guarantees and can be resource hog for multi-tenant environment.
- Datacenters consists of machines with varied capacities and characteristics (unless you are on cloud)
- Heterogeneous workload + ephemeral environment doesn't make it easy
- Whatever we do, remember that effective workload management remains a difficult challenge

Pre-requisite

- The first pre-requisite is: ISOLATION
- Isolation can be achieved through containers.
- Isolation of process space, file-system, network stack, user namespace, disk usage, disk IO, network bandwidth etc.
- With Mesos, you can do it with Mesos/Unified containerizer or Docker containerizer

Mesos vs Docker containerizer



Pros/Con

Mesos/Unified containerizer	Docker containerizer
Pros	**Pros**
• fine grained operating system controls e.g., cgroups & namespaces	• Standard way of orchestrating docker containers through Mesos
• Already provides custom isolators like disk quota, network performance & segregations	• Battle tested. It just works with scale.
• Pick and choose which isolators you want while container initialization	
• Easily extensible with custom isolators	
Con	**Con**
• Cannot leverage additional features of docker-engine like ps, logs, exec, inspect etc.	• Need to maintain docker-engine on every mesos agent.
	• And when you upgrade docker-engine, tasks die
	• Only provides CPU and Memory isolations

Assumption

- Don’t get bogged down with the details of each
- Whatever containerizer you choose, trust me there is a lot of work ahead
- Production containerized workload isn’t as simple as spinning up a container and you are over with
- And the problem is if you have sold the Mesos idea too much in your company, they will come haunt you back
  - They need this
  - And that?
  - And what about that?
  - And how can I do that?

Use-cases

- Let's switch the gear from system (mesos) level to user (application) level.
- How do you support their myriad range of apps that have unique use-cases ?
- How do you provide a common platform that all these apps can leverage ?
- None of containerizer provides functionalities out of the box that you need to support these heterogeneous apps on a single cluster.

Issue 1:

Structured Logging

Structured Logging

- By default, Mesos just stores the STDOUT/STDERR of the containers in plain text in the sandbox.
- And it just piles up.
- With the newer Mesos 0.27, it lets you do logrotate on those files with "LogrotateContainerLogger" module.
- And I guess that's about it.
- This may not be sufficient for some apps to point all the output to STDOUT/STDERR.
- What if they generate multiple log files and want to keep them separate?
- What if your app generates binary data in logs?

Structured Logging

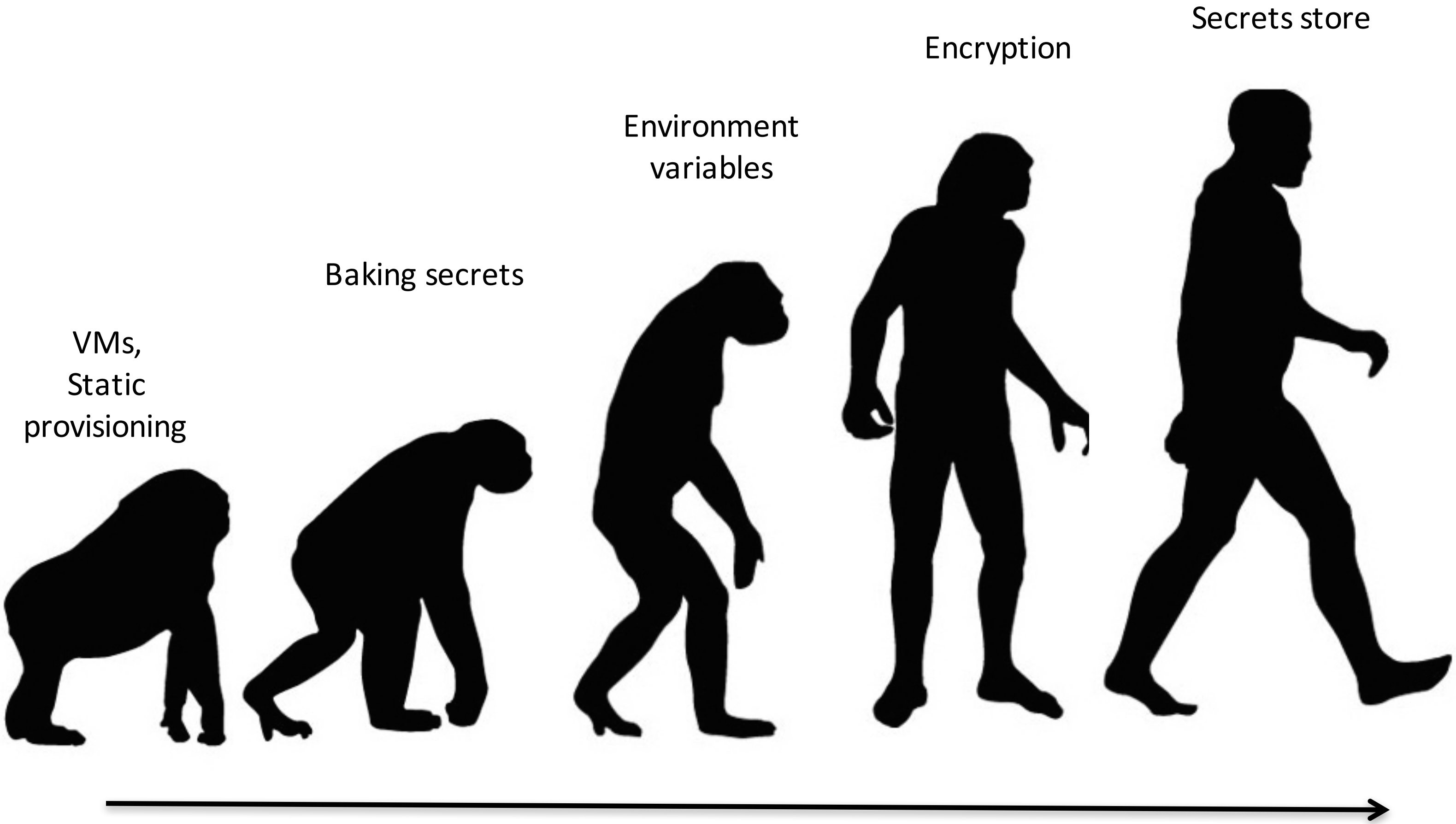
- There is no structured solution for logs with Mesos.
- Apps have varied use cases for logs
  - Some want to index in Elasticsearch
  - Some want to persist in object storage
  - Some want to run analytics in real time on grid
- In short, logs have to shipped away
  - Either to centralized logging
  - Either to message queues
  - Either to stream processing platform like Riemann or Graylog for real-time metrics analysis
- So you have to provide a solution that covers all these use-cases

Issue 2:

Application Secrets

Application secrets

- One of the main things while running containerized workload is how do you deal with secrets
- Secrets are important. More important how to properly secure them in containerized envt.
- Some of the secrets that you may need are:
  - Database credentials
  - API tokens
  - TLS certificates/keys
  - GPG keys
  - SSH keys



VMs,
Static
provisioning

Baking secrets

Environment
variables

Encryption

Secrets store

Fig1: Evolution 5 (Parry, 2012).

Application secrets

- Again docker doesn't have any solution. They have tons of PRs: (Stijn, 2015).
  - Add private files support #5836
  - Add secret store #6075
  - Continuation of the docker secret storage feature #6697
  - The Docker Vault" #10310
  - Provide roadmap / design for officially handling secrets. Make injecting secrets pluggable, so that they use existing offerings in this area, for example: Vault, Keywiz, Sneaker
- Solution should be how we can pass application secrets dynamically during container runtime.
- I gave a talk about that at USENIX and SCALE.
  - More details at: http://elasticcompute.io

Issue 3:

Application Configuration Management

Application Config Management

- All the heterogeneous apps now need a common place from where they can pull their configs
- We can’t let different apps pulling configs from various stores.
  - Integration with Mesos could become challenging for some
- And we cant be always be baking configs in images
- And all the same issues discussed in previous slide applies
- Also, if you want to make your environment really dynamic:
  - you should be able to change configs in the containers on the fly
  - and reload them

Issue 4:

Running databases

Run databases

- Besides running frontends, there is a genuine need to run backend databases with Mesos
- Problem is containers are ephemeral
- So to achieve persistence, databases should be run on some shared storage like NFS or through some mounts
- If it is NFS, then the volume is exposed to all the hosts in the cluster.
- But if I am running it through NFS, what is the need to run from Mesos?

Run databases

- Solution is to use a block device that provides one to one container mapping
- If it is one to one container mapping, what happens when the container goes away?
- The new container should be able to use the block device again
- So, the solution should be able to:
  - Mount - Locks, Maps and Mounts Block Device to the Host system
  - Unmount - Unmounts, Unmaps and Unlocks the Block Device on request
- We have created this docker plugin for CEPH DFS: https://github.com/yp-engineering/rbd-docker-plugin
- It is now an official block device solution with CEPH recommends
- Our idea is to use the same commodity hardware for carving out block devices from the cluster

Issue 5:

Integrations with hardware load balancers

Integrations with hardware load balancers

- All the other software service discovery mechanism like: mesos-dns, Consul, Bamboo/HAProxy or Traefik are good
- Some of them are buggy, WIP, limited feature set and haven't been tested at production workload
- When it comes to supporting production workload, there is an official need to integrate with existing hardware load-balancer
- They provide robust features like: websockets, SSL termination, custom health checks, fancy graphs etc.
- More so, you company has heavily invested in them already ;-)

Integrations with hardware load balancers

- Typically, we statically configure members for a VIP in hardware LB
- In this ephemeral containerized envt, there is a need to update the members of the pool dynamically
- So, they should be integrated to listen to the change of state of the cluster

Issue 6:

Isolating resource hogs

Isolating resource Hogs

- Some apps are I/O or network intensive
- They have a tendency to starve or severely affect other apps running on the same host
- If you don't have proper isolations, you can't effectively run heterogeneous workload
- For network traffic control:
  - MesosContainerizer: already an isolator for "net_cls"
  - Docker: you have to pass a "cgroup net_cls" option
- For Disk I/O control:
  - MesosContainerizer: Nothing right now
  - Docker: docker-1.10 has added options to control disk I/O
    - --device-read-bps, --device-write-bps, --device-read-iops, --device-write-iops, and --blkio-weight-device
- For now, we have profiled those apps and run them on dedicated resources (semi static partitioning)

Other lessons learned

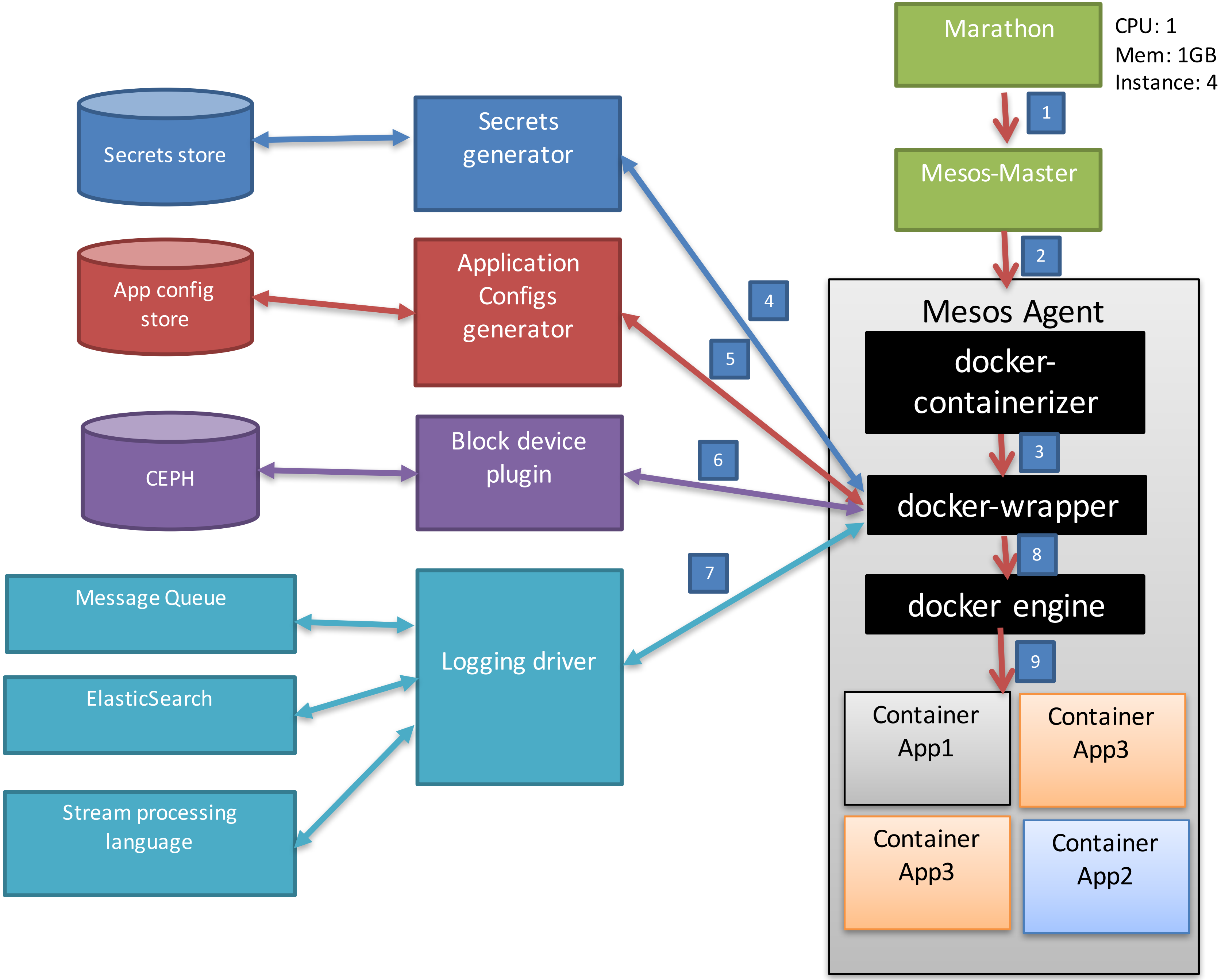
Other lessons learned

- Troubleshooting apps in production
  - Enable ssh access?
  - How to be SOX/PCI compliance?
- If all the dev teams want full control, do you run single cluster or multiple small clusters?
- Remember the 80/20 rule — 80% of the performance improvement comes from tuning the application, and the rest 20% comes from tuning the infrastructure components.
- Now that all things are HA, you should seriously consider if you need underlying RAID config on your machines

Other lessons learned

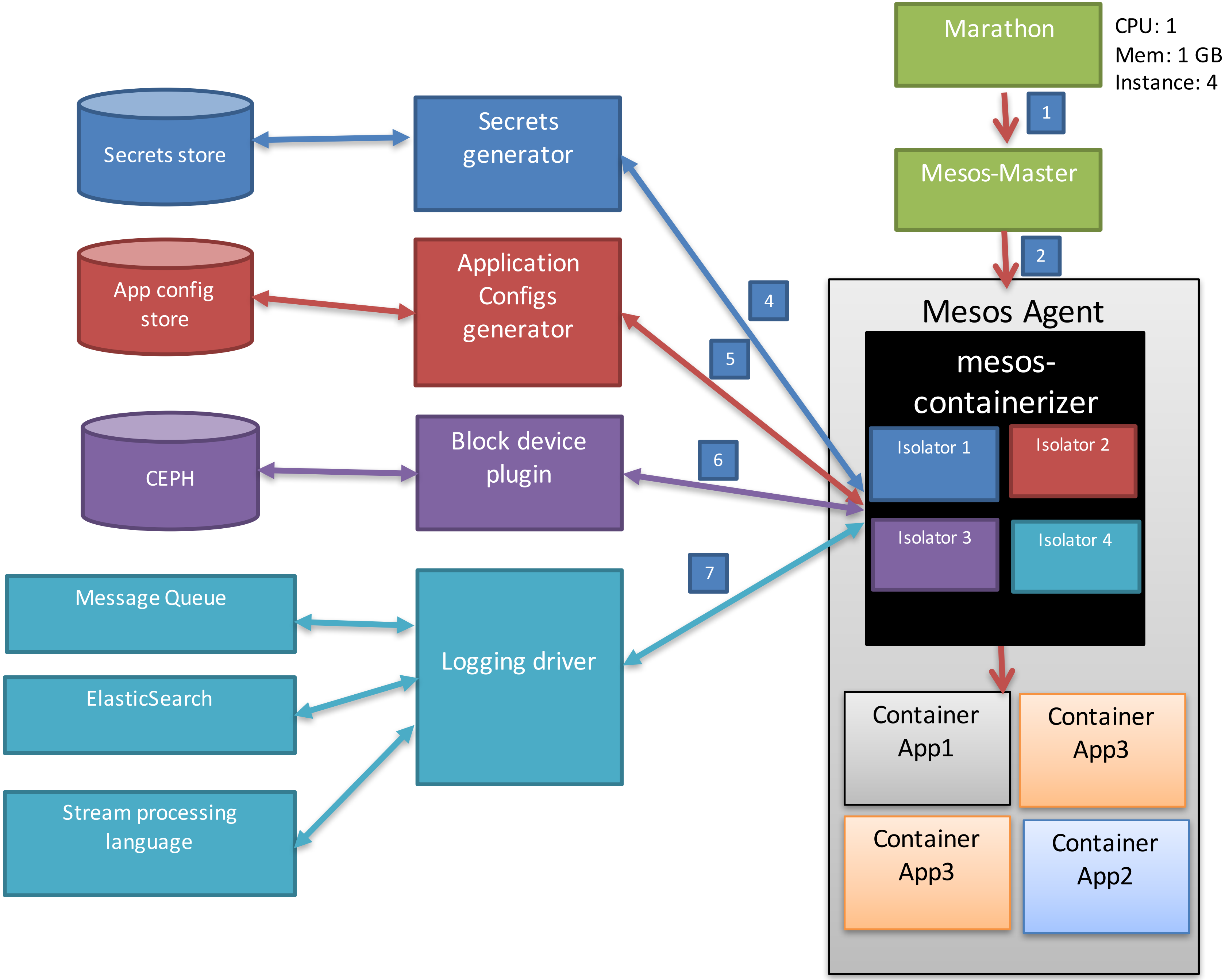
- Lots of new components
- Setup for any new component is easy but running them in production is different
- Simulate any new component with a peak load
- Monitor and alert on every entry and exit endpoint. Monitor for thresholds.
- Lots of floating pieces. Needs to be anchored.
- All the components need to be thoroughly understood
- Administration

Our solution



MesosCon 2016

What should be optimal solution ?


Marathon
CPU: 1
Mem: 1 GB
Instance: 4
1
Mesos-Master
2
Mesos Agent
mesos-containerizer
Isolator 1
Isolator 2
Isolator 3
Isolator 4
Container App1
Container App3
Container App3
Container App2
Secrets store
Secrets generator
App config store
Application Configs generator
CEPH
Block device plugin
Message Queue
ElasticSearch
Stream processing language
Logging driver
4
5
6
7

Conclusion

- Docker is slow
- I don’t want to wait on what they are going to release
- And some of the extensions they provide aren’t really modular
- It’s like my way or highway
- I have an immediate need of new features now
- I have an immediate need to support my heterogeneous workload that has varying needs

Conclusion/Preaching

- Have to build these extensions using Isolator modules or hooks within Mesos
- Stop treating containerization as a second class citizen within Mesos
- Stop saying containerization serves as one of its goals
- Sooner or later, everything would be running in containers
- Mesos should be a solid orchestrator covering most of the use-cases that we discussed today
- Because if it doesn't, Mesos will just end up being a resource manager and scheduler
- And in the end, it will be running other orchestrators like kubernetes and swarm as framework on Mesos

What are we doing at YP Engineering?

- We are doing all these crazy stuff you saw earlier
- Building, managing and running them at scale
- Open source contribution:

  www.github.com/yp-engineering

REFERENCE LIST

- Parry, Wynne. (2012). File:human-evolution.jpg.[Image file]. Retrieved from: http://www.livescience.com/images/i/000/025/831/original/human-evolution.jpg?1332952687
- Stijn, Sebastiaan. (2015). Secrets: write-up best practices, do's and don'ts, roadmap #13490. Retrieved from: https://github.com/docker/docker/issues/13490
- Docker: http://www.docker.com
- Mesos: http://mesos.apache.org
- Kubernetes: http://kubernetes.io
- Swarm: https://docs.docker.com/swarm/
- Vault: https://www.vaultproject.io/
- Keywhiz: http://square.github.io/keywhiz/
- Sneaker: https://github.com/codahale/sneaker
- CEPH DFS docker-plugin: https://github.com/yp-engineering/rbd-docker-plugin
- Mesos-dns: https://github.com/mesosphere/mesos-dns
- Consul: http://consul.io
- Traefik: https://traefik.io/
- Bamboo/HAProxy: https://github.com/QubitProducts/bamboo
- Elasticsearch: https://www.elastic.co/
- Riemann: http://riemann.io/
- Graylog: https://www.graylog.org/
- CEPH: http://ceph.com/

Thank you for listening !!

Q/A

Imran Shaikh
Lead/Architect
yp

Blog http://elasticcompute.io
@imranshaikh
imran@elasticcompute.io